▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 22

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 22

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-159-2
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

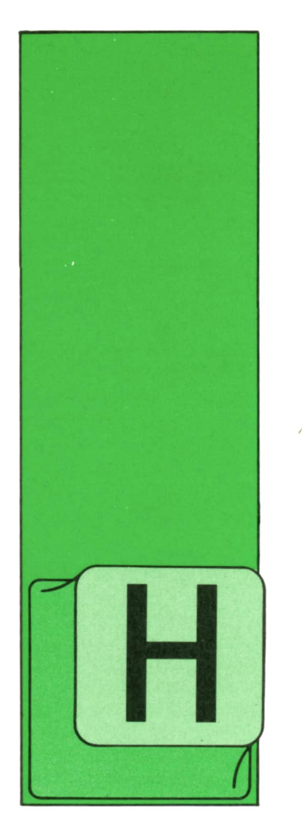
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

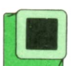
Agosto, 1987.
P.V.P. Canarias: 335,–.

INDICE



INFORMATICA BASICA

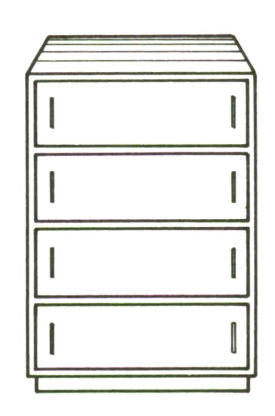
LA MEMORIA



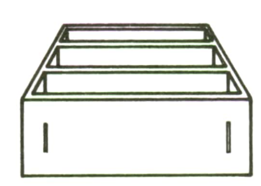
Introducción

HASTA el momento hemos visto mecanismos capaces de realizar operaciones. Hemos visto también que con una hábil combinación de éstos, podemos realizar todas las operaciones lógicas.

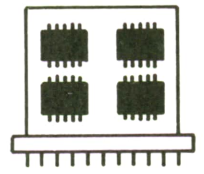
Pero toda operación necesita operandos, y éstos, necesitan almacenarse en algún sitio. También es necesario almacenar todos los bits que utiliza el ordenador para realizar las operaciones. Todo este espacio de almacenamiento necesario para retener el conjunto de informaciones se denomina **memoria.**



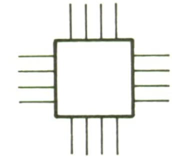
1.ª GENERACION
Un armario de circuitos con válvulas electrónicas



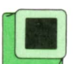
2.ª GENERACION
Un cajón con placas transistorizadas



3.ª GENERACION
Aumenta la capacidad de integración



4.ª GENERACION
Integración a gran escala en una cápsula



Tipos de memorias

Podemos hacer una primera clasificación de las memorias en base a su funcionamiento.

— Memorias no volátiles: son memorias que no pierden la información almacenada en ellas aunque se quite la alimentación.

— Memorias volátiles: se utilizan para almacenar datos de forma temporal. La memoria residente en la CPU es de este tipo, ya que al desenchufar el ordenador los programas y datos almacenados en él se borran. En un principio estaban hechas con núcleos de ferrita, actualmente se utilizan semiconductores principalmente.

Existen dos tipos:

— Memorias **ROM.** Son memorias de solo lectura.

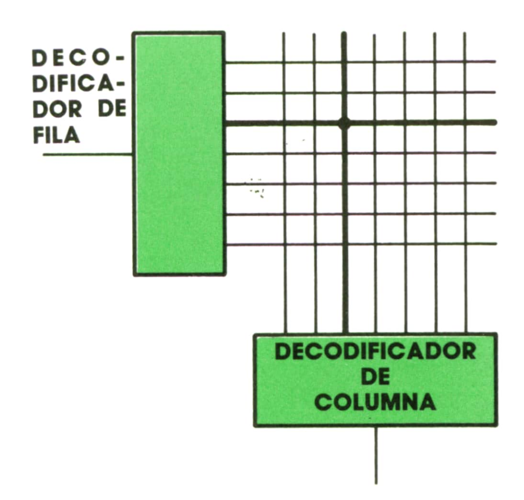
• ROM: la información que contienen

viene dada por el fabricante; es la más utilizada en la fabricación.

• PROM: son programables por el usuario, pero una vez que han sido programadas no pueden cambiar su contenido.

• EPROM: son programables también por el usuario pero pueden cambiar su contenido; son las más utilizadas.

• EAPROM: programables por el usuario pero alterables eléctricamente. Su diferencia con las memorias RAM es que tienen mayor tiempo de acceso.




Mediante los decodificadores es posible localizar cualquier bit en el nudo de la red.

— Memorias **RAM.** Son memorias de lectura y escritura, el tiempo de acceso es el mismo en ambas operaciones.

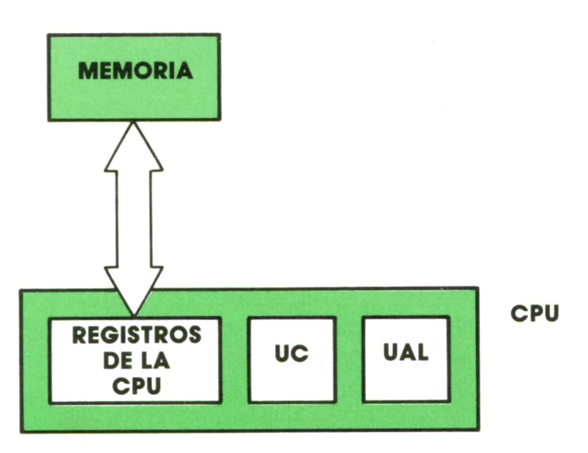
• Estáticas: no necesitan depender de los flancos de las señales producidas por el reloj del sistema. No pierden información.

• Dinámicas: necesitan una lógica de control que produzca flancos y períodos de refresco. Almacenan información durante un tiempo determinado.

• Seudoestáticas: son estáticas de cara al usuario, internamente son dinámicas (tienen una lógica de control).

Una segunda clasificación, más conocida quizá por personas sin grandes conocimientos de electrónica, atiende a su situación.

El concepto de memoria se aplica indistintamente tanto a la residente en la CPU, como a cualquier dispositivo de almacenamiento. Sin embargo, existe una diferencia fundamental entre las dos clases: la memoria auxiliar no puede ser ejecutada directamente por la CPU, por lo que para poder ejecutar cualquier programa almacenado en este tipo de dispositivos (discos, cintas, etc.), habría que cargarlo antes en la memoria principal.




Diferencia entre la memoria y los registros de la CPU.

Las memorias que no están residentes en el ordenador se denominan **memorias de masa,** y se pueden clasificar en dos grandes grupos: secuenciales y de acceso directo o aleatorio.

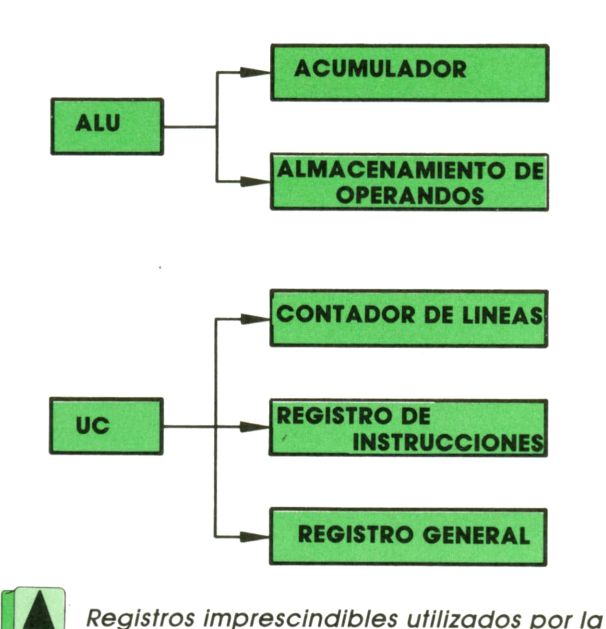
Las memorias de masa de acceso secuencial son aquellas a las que se accede a un dato de forma que antes se ha tenido que pasar por todos los que le preceden. Los ejemplos más conocidos de dispositivos de almacenamiento de este tipo son las cintas perforadas y las cintas magnéticas.

Los dispositivos de almacenamiento de acceso directo tienen la ventaja de poder localizar cualquier dato de forma directa; los discos, los disquetes y tambores magnéticos son los más utilizados.

 La memoria en la CPU

Vamos a referirnos ahora a la memoria que se encuentra en la CPU. Esta memoria está construida totalmente con elementos electrónicos, es muy rápida pero tiene el inconveniente de que es volátil.

Dentro de la memoria de la CPU podemos establecer una separación en dos bloques.



Registros imprescindibles utilizados por la CPU.

Uno de estos bloques es el constituido por un conjunto de registros alojados en el interior del microprocesador. Estos registros son utilizados por la Unidad de Control y la Unidad Aritmético-Lógica para almacenar la información que se está tratando en un momento dado. Este bloque tiene una importancia fundamental al ser determinante en la velocidad de trabajo del microprocesador. Cuanto más generoso sea el sistema en dotarle de registros, mayor será la capacidad de trabajo y la posibilidad de ser rápido.

En líneas generales, el número de registros de la CPU depende del número de recursos propios del sistema, pero, desde luego, hay registros de los que no se puede prescindir.

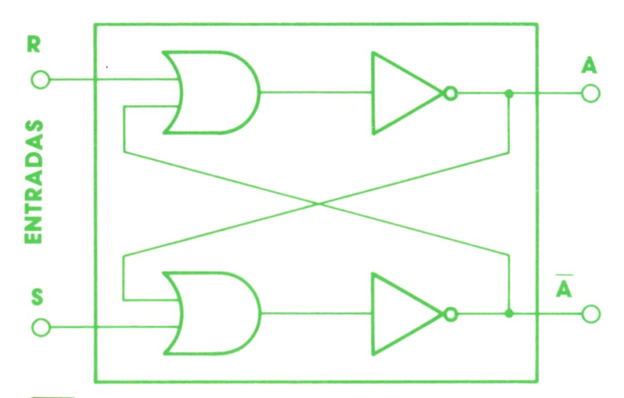
La Unidad Aritmético-Lógica necesita al menos de un registro, llamado **acumulador,** donde mantienen el resultado de las operaciones que realiza, además de uno o dos registros más donde sitúa los operandos. La Unidad de Control necesita un registro que utiliza como **contador** de las líneas de programa, en concreto, le sirve para almacenar el número de la siguiente instrucción a ejecutar; necesita, además, el **registro de instrucción** donde retienen los códigos de las operaciones o instrucciones posibles y que utiliza para decodificarlas convenientemente; y finalmente, un registro general donde se almacenan temporalmente operandos o direcciones de memoria concretas.

Existen otros registros que mantienen valores particularmente significativos, como el **registro de estado,** en el que se encuentran las direcciones de programa desde las que se han ejecutado los saltos o subrutinas.



Almacenamiento en memoria

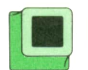
Veamos, ahora, los dispositivos que hacen posible mantener un valor 0 ó 1 mientras no se decide que cambie. En el resto de dispositivos, los valores sólo se mantienen mientras dura el paso de corriente, desapareciendo después sin dejar señal del valor que se había tenido.





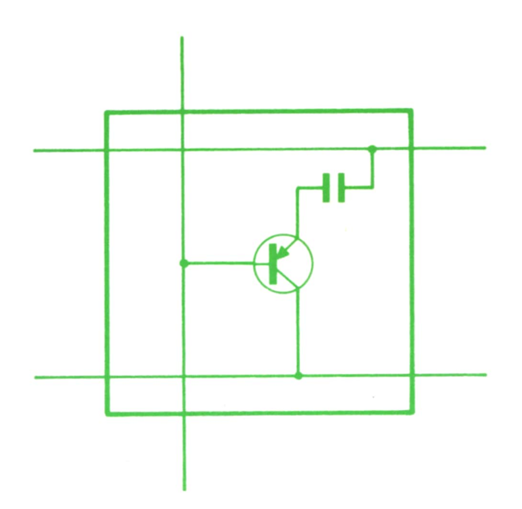
Representación del biestable por medio de puertas lógicas.

El dispositivo capaz de «recordar» el valor anterior es el llamado **biestable** RS; su funcionamiento consiste en «realimentar» la entrada con el valor que tenía a la salida, mientras no haya una alteracion en las entradas R y S, no variará la salida. Por tanto, estará recordando continuamente el estado en el que se encuentra; es capaz de «almacenar un bit».



La memoria central

La memoria central es la que apoya a la CPU. Entre ésta y los registros existe una diferencia relativa a la organización, tamaño e incluso construcción de los dispositivos básicos.




Unidad elemental de memoria.

La memoria central es destinada al almacenamiento de la información con la que se ha de trabajar en un proceso cualquiera. Entre esta información están las instrucciones acerca del trabajo. Sólo una cantidad muy pequeña de esta información ocupa los registros en un instante preciso.

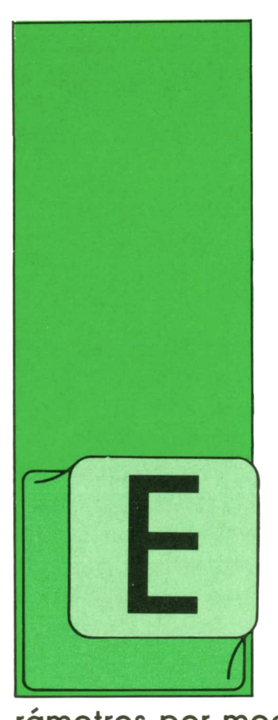
 La memoria virtual

La memoria virtual es una facilidad que se da al programador de forma que aparentemente puede utilizar mucha más memoria de la que en realidad existe. De esta manera los programadores pueden ejecutar sus programas sin que realmente quepan en la memoria del ordenador. La memoria virtual se apoya en la memoria real para poder moverse, es decir, se introduce en la memoria real, en particiones, ya que por ser mayor no cabría, y así en sucesivas entradas y salidas puede realizar sus funciones.

MAQUINA 8088



Ejemplo de uso de rutinas



L ejemplo que proponemos nos va a servir de pretexto para analizar con detalle un programa en el que se hace un extenso uso de rutinas. En ellas se pasan parámetros por medio de los registros que es el método más fácil y más usado. Como casos especiales, se incluyen rutinas llamadas por otras rutinas y otras cuyos puntos de entrada se definen por medio de la instrucción LABEL.

El programa PROG4 lee del teclado dos números hexadecimales, calcula su suma y representa el resultado también en hexadecimal, por lo que, además de instructivo, puede resultar útil.

El listado contiene 7.500 caracteres, de los cuales, sólo 1.500 corresponden a instrucciones del programa y el resto son comentarios que acompañan a todas las líneas. La primera parte del comentario es un número secuencial que utilizaremos para hacer referencia a las líneas.

PROG4 está compuesto por un único segmento llamado **CODIGO,** cuyos límites están definidos en las líneas 003 y 129. En el segmento se han definido cuatro procedimientos:

PROC1. El el procedimiento que hace las veces de programa principal. En efecto, recibe control del DOS, tal como se especifica en la instrucción END (línea 130), prepara su retorno en las líneas 006 a 008, y vuelve al DOS en la instrucción RET de la línea 038. Este procedimiento es el único que se ha definido de tipo FAR, por requerirlo así el método que se emplea para volver al DOS.

PROC2. Este procedimiento contiene en su interior instrucciones LABEL que definen los puntos de entrada a las rutinas OBTENER, REPRESENTA2 y REPRESENTA1.

OBTENER sirve para leer dos caracteres del teclado y devolver al programa que lo llamó (PROC1) el código binario correspondiente. Este rutina llamaba a HEXA-BIN para realizar la conversión. La lectura del teclado se realiza mediante la llamada número 1 al DOS, que se explicará más adelante cuando se trate de las interrupciones por software.

REPRESENTA2 sirve para representar en pantalla los dos caracteres ASCII definidos en los registros AH y AL. Para ello llama dos veces a REPRESENTA1.

REPRESENTA1 sirve para representar en pantalla el carácter ASCII definido en el registro DL. Lo cual se hace mediante la llamada número 2 al DOS.

BIN_HEXA. Que sirve para obtener los dos caracteres hexadecimales ASCII que representan al byte que en el momento de la llamada está contenido en el registro AL. En el programa se la llama en las líneas 027 y 031 para convertir los dos bytes que resultan de la suma pedida.

HEXA_BIN. Es una rutina que devuelve, en el registro AL, el número binario correspondiente a los dos caracteres hexadecimales ASCII que en el momento de la llamada están definidos en los registros AH y AL. Es, por tanto, la rutina inversa de la anterior y se la llama desde la rutina OBTENER (línea 052).

Una vez conocidas las 5 rutinas, la lectura del programa principal resulta muy sencilla. En efecto, el programa se puede resumir en lo siguiente:

1. Prepara la vuelta al DOS.
2. Escribe en la pantalla el carácter que indica que el programa está preparado para responder.
3. Obtiene del teclado dos caracteres ASCII que generan el primer sumando.
4. Si algún carácter no es válido, vuelve al DOS.

5. Escribe en la pantalla el signo +.
6. Obtiene del teclado dos caracteres ASCII que generan el segundo sumando.
7. Si algún carácter no es válido, vuelve al DOS.
8. Escribe en la pantalla el signo =.
9. Suma los dos sumandos obtenidos.
10. Convierte el resultado a hexadecimal ASCII.
11. Escribe en la pantalla la parte más significativa de la suma.
12. Escribe en la pantalla la parte menos significativa de la suma.
13. Vuelve al punto 2.

```
            NAME     PROG4            ;001  Programa PROG4. Suma de hexadecimales.
;-----------------------------------  ;002  ------------------------------------
CODIGO      SEGMENT                   ;003  Comienzo del Segmento CODIGO.
            ASSUME   CS:CODIGO        ;004  Supone que CS direcciona el segmento CODIGO.
PROC1       PROC     FAR              ;005  Comienza el procedimiento PROC1 (principal).
            PUSH     DS               ;006  Prepara en el STACK
            MOV      AX,0             ;007  la vuelta al DOS para cuando
            PUSH     AX               ;008  termine el programa.
L1:                                   ;009
            MOV      DL,'+'           ;010  Representa en pantalla
            CALL     REPRESENTA1      ;011  el carácter (+).
            CALL     OBTENER          ;012  LLama a la rutina OBTENER (primer sumando).
            JC       FIN              ;013  Va a FIN si el carry flag=ON, es decir, si
                                      ;014  se ha tecleado algún carácter inválido.
            PUSH     AX               ;015  Salva en la pila el primer sumando.
            MOV      DL,'+'           ;016  Representa en pantalla
            CALL     REPRESENTA1      ;017  el caracter separador de los sumandos (+)
            CALL     OBTENER          ;018  LLama a la rutina OBTENER (segundo sumando).
            POP      BX               ;019  Recupera de la pila el primer sumando.
            JC       FIN              ;020  Va a FIN si el carry flag=ON, es decir, si
                                      ;021  se ha tecleado algún carácter inválido.
            MOV      DL,'='           ;022  Representa en pantalla
            CALL     REPRESENTA1      ;023  el caracter (=).
            ADD      BX,AX            ;024  Suma (en BX) los 2 números tecleados. <----
                                      ;025
            MOV      AL,BH            ;026  Convierte a hexadecimal (ASCII) el
            CALL     BIN_HEXA         ;027  byte más significativo de la suma (BH) y
            CALL     REPRESENTA2      ;028  lo representa en pantalla.
                                      ;029
            MOV      AL,BL            ;030  Convierte a hexadecimal (ASCII) el
            CALL     BIN_HEXA         ;031  byte menos significativo de la suma (BL) y
            CALL     REPRESENTA2      ;032  lo representa en pantalla.
                                      ;033
            MOV      AX,0D0AH         ;034  Representa en pantalla
            CALL     REPRESENTA2      ;035  los caracteres (0D y 0A) de salto de línea.
            JMP      L1               ;036  Va a L1 para repetir el proceso.
FIN:                                  ;037
            RET                       ;038  Termina del programa. Vuelve al DOS.
PROC1       ENDP                      ;039  Fin del procedimiento PROC1 (principal).
;-----------------------------------  ;040  ------------------------------------
PROC2       PROC     NEAR             ;041  Comienzo del procedimiento PROC2.
OBTENER     LABEL    NEAR             ;042  Punto de entrada de la rutina OBTENER.
; Esta rutina devuelve en el registro AL el número binario
; correspondiente a dos caracteres hexadecimales tecleados.
            MOV      AH,1             ;045  Mueve 1 a AH (para especificar DOS CALL n.1)
            INT      21H              ;046  DOS CALL n.1 (Lectura del teclado sobre AL).
            PUSH     AX               ;047  Salva el primer caracter obtenido (AL)
            MOV      AH,1             ;048  Mueve 1 a AH (para especificar DOS CALL n.1)
            INT      21H              ;049  DOS CALL n.1 (Lectura del teclado sobre AL).
            POP      BX               ;050  Recupera el primer caracter en BL, lo
            MOV      AH,BL            ;051  copia en AH (ya se tiene AH-AL) y
            CALL     HEXA_BIN         ;052  llama a HEXA_BIN para convertirlos a binario
            RET                       ;053  Vuelve al programa llamante. Resultado en AL
;-----------------------------------  ;054  ------------------------------------
REPRESENTA2          LABEL NEAR       ;055  Punto de entrada de la rutina REPRESENTA2.
; Esta rutina representa en pantalla los caracteres recibidos en AH y AL.
            MOV      DL,AH            ;057  Mueve a DL el primer caracter (AH).
            CALL     REPRESENTA1      ;058  LLama a REPRESENTA1.
            MOV      DL,AL            ;059  Mueve a DL el segundo caracter (AL).
            CALL     REPRESENTA1      ;060  LLama a REPRESENTA1
            RET                       ;061  Vuelve al programa llamante.
;-----------------------------------  ;062  ------------------------------------
REPRESENTA1          LABEL NEAR       ;063  Punto de entrada de la rutina REPRESENTA1.
; Esta rutina representa en pantalla el caracter recibido en DL.
            PUSH     AX               ;065  Salva el registro AX.
            MOV      AH,2             ;066  Mueve 2 a AH (para especificar DOS CALL n.2)
            INT      21H              ;067  DOS CALL n.2 (Representa DL en pantalla).
            POP      AX               ;068  Recupera el registro AX.
            RET                       ;069  Vuelve al programa llamante.
PROC2       ENDP                      ;070  Fin del procedimiento PROC2.
;-----------------------------------  ;071  ------------------------------------
BIN_HEXA             PROC NEAR        ;072  Punto de entrada de la rutina BIN_HEXA.
; Esta rutina devuelve en el registro AX los dos caracteres hexadecimales que
; representan al byte que en la llamada se había recibido en AL.
            PUSH     CX               ;075  Salva en la pila el registro CX.
            MOV      AH,0             ;076  Mueve 0 a AH.
            MOV      CL,4             ;077  Define el contador de desplazamiento=4 bits.
```

```
        SHL     AX,CL           ;078  Desplaza AX 4 bits izq. (00 xy) --> (0x y0)
        SHR     AL,CL           ;079  Desplaza AL 4 bits dcha.(0x y0) --> (0x 0y)
        ADD     AX,'00'         ;080  Suma código ASCII del origen de los números.
                                ;081
        CMP     AH,'9'          ;082  Compara AH con el ASCII '9'.
        JLE     LBL1            ;083  Va a LBL1 si AH es menor o igual.
        ADD     AH,7            ;084  Suma 7 a AH. (porque hay 7 caracteres ASCII
LBL1:                           ;085              extras entre el '9' y la 'A')
        CMP     AL,'9'          ;086  Compara AL con el ASCII '9'.
        JLE     LBL2            ;087  Va a LBL2 si AL es menor o igual.
        ADD     AL,7            ;088  Suma 7 a AL. (porque hay 7 caracteres ASCII
LBL2:                           ;089              extras entre el '9' y la 'A')
        POP     CX              ;090  Recupera de la pila el registro CX.
        RET                     ;091  Vuelve al programa llamante.
BIN_HEXA        ENDP            ;092  Fin de la rutina BIN_HEXA.
;------------------------------ ;093 ---------------------------------------
HEXA_BIN        PROC NEAR       ;094  Punto de entrada de la rutina HEXA_BIN.
; Esta rutina comprueba si los dos bytes de AX (AH, AL) contienen caracteres
; hexadecimales, es decir, si pertenecen al juego: 0123456789ABCDEF
; - En caso afirmativo devuelve carry-flag=OFF, AH=00 y AL=byte convertido.
; - En caso negativo   devuelve carry-flag=ON  (caso erróneo).
        PUSH    CX              ;099  Salva en la pila el registro CX.
        PUSH    BX              ;100  Salva en la pila el registro BX.
        MOV     CX,2            ;101  Define el contador de bucles=2.
LBL3:                           ;102  Comienzo del bucle.
        SUB     AL,'0'          ;103  Resta ASCII '0' de AL y compara resta con 0.
        JL      LBL6            ;104  Va a LBL6 si es menor
        CMP     AL,10           ;105  Compara AL con 10.
        JL      LBL4            ;106  Va a LBL4 si es mayor.
        SUB     AL,7            ;107  Resta 7 de AL (hay 7 ASCII entre '0' y 'A').
        AND     AL,0DFH         ;108  Anula un bit. Pasa minusculas a mayusculas.
        CMP     AL,10           ;109  Compara AL con 10.
        JL      LBL6            ;110  Va a LBL6 si es menor.
        CMP     AL,15           ;111  Compara AL con 15.
        JG      LBL6            ;112  Va a LBL6 si es mayor.
LBL4:                           ;113  Ahora se tiene el código binario en AL.
        XCHG    AL,AH           ;114  Intercambia AL (el código calculado) con AH.
        LOOP    LBL3            ;115  Fin del bucle. (repetir con el otro byte).
        MOV     CL,4            ;116  Define contador de desplazamiento=4 bits.
        SHL     AL,CL           ;117  Desplaza AL 4 bits izq. (0x 0y) --> (0x y0).
        SHR     AX,CL           ;118  Desplaza AX 4 bits dcha.(0x y0) --> (00 xy).
        CLC                     ;119  Define el flag de carry=OFF (no hay error).
LBL5:                           ;120
        POP     BX              ;121  Recupera de la pila el registro BX.
        POP     CX              ;122  Recupera de la pila el registro CX.
        RET                     ;123  Vuelve al programa llamante(resultado en AL)
LBL6:                           ;124  Aquí llega en caso de caracter erróneo.
        STC                     ;125  Define el flag de carry = ON  (hay error).
        JMP     LBL5            ;126  Va a LBL5 para terminar la rutina.
HEXA_BIN        ENDP            ;127  Fin de la rutina HEXA_BIN.
;------------------------------ ;128 -------------------------
CODIGO  ENDS                    ;129  Fin del segmento CODIGO.
        END     PROC1           ;130  Fin del programa fuente. (Ejecutar PROC1).
```

Se recuerda que para ejecutar el prógrama son necesarios los siguientes pasos:

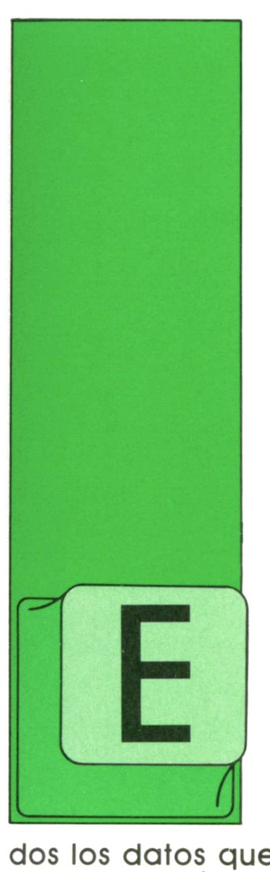
- Teclear el programa fuente desde un editor. (Se puede prescindir de los comentarios.)
- Generar el módulo objeto. (Mediante A>MASM PROG4;)
- Generar el módulo ejecutable. (Mediante A>LINK PROG4;)
- Ejecutar el programa generado. (Mediante A>PROG4)

La generación del módulo ejecutable mediante LINK, da un mensaje en el que avisa de un error. Esto es debido a que no se ha definido el segmento de STACK, pero no debe preocuparnos, porque el DOS asigna por omisión un área que no causa ningún problema.

Por último, en la ejecución del programa PROG4 observaremos que aparece un símbolo especial que significa «preparado» y el programa queda a la espera de que se defina un primer sumando, tecleando dos caracteres hexadecimales. Una vez tecleados, el programa escribe el signo más (+) y espera que se defina el segundo sumando. Al hacerlo, el programa escribe el signo igual (=), a continuación el valor de la suma, también en hexadecimal, y por último escribe el signo de «preparado» en la línea siguiente para continuar el proceso. El programa termina cuando alguno de los caracteres tecleados no es un dígito hexadecimal.

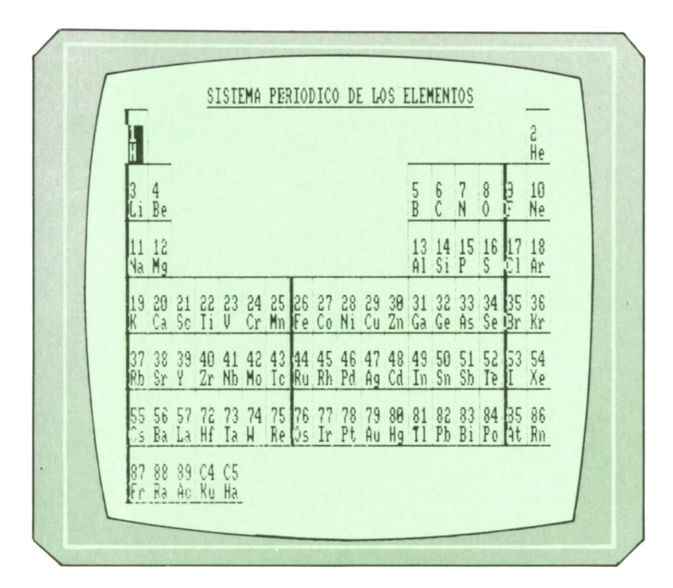
PROGRAMAS

Programa: Base de datos química para AMSTRAD



STE programa es una auténtica base de datos con todos los elementos químicos conocidos. Con esta base de datos podremos saber, en cualquier momento, todos los datos que necesitemos sobre un elemento químico cualquiera.

Una vez introducido el programa, y tras hacer un RUN nos aparece en la pantalla una tabla periódica tal y como suelen hacer aparecer en los libros.



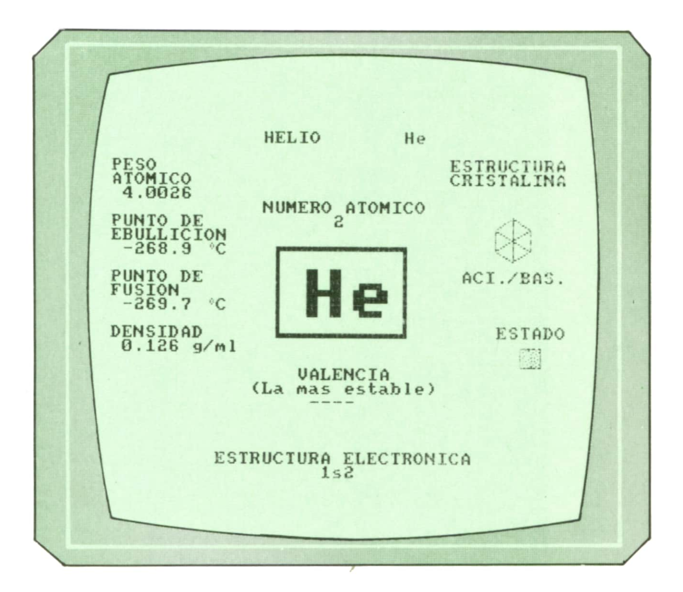


La tabla periódica tal y como aparece en la pantalla.

Una vez que nos aparece dicha tabla en la pantalla, podemos utilizar las teclas del cursor para movernos de un elemento a otro. Sabremos la posición del cursor porque el elemento sobre el que se coloque se nos aparecerá invertido.

Tras mover el cursor al elemento del cual queremos saber todos los datos que almacena el programa, para ver dichos datos sólo tenemos que pulsar la tecla ENTER.

En la figura 2 podemos ver todos lo datos que tiene almacenado el programa sobre el HELIO.





Información que contiene el programa sobre el HELIO.

En general, la información que almacena este programa es la siguiente:

- — Nombre del elemento.
- — Siglas del elemento
- — Número atómico del elemento.
- — Valencias más estables.
- — Peso atómico.
- — Punto de ebullición en grados centígrados.

— Punto de fusión en grados centígrados.

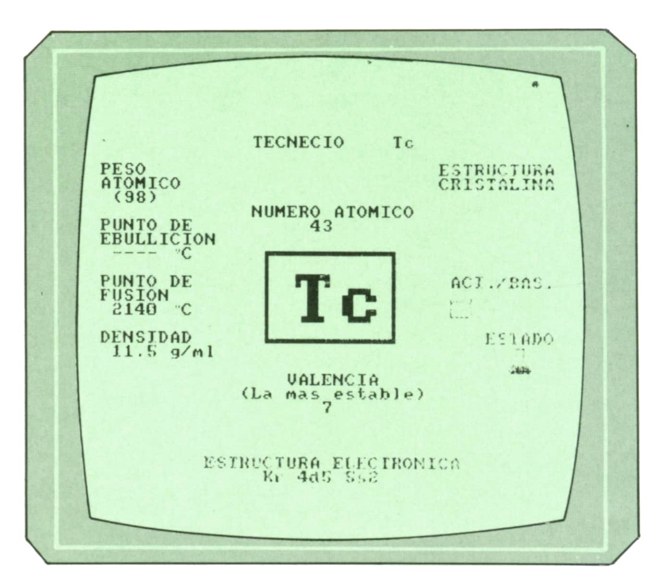
— Densidad en gramos/milímetro.

— Estructura electrónica.

— Estructura cristalina. Nos dibuja su estructura. Esta puede ser tetraédrica, tetragonal, cúbica simple, cúbica con centro en una cara, cúbica con centro en el interior, rómbica, romboédrica, monocíclica y hexagonal.

— Estado. Nos dice si es gas, sólido o líquido mediante un dibujo distinto. En el caso de un gas nos aparecerá un cuadrado con puntitos, los sólidos como un rectángulo y los sólidos como un matraz lleno de líquido.

— Acidez/basicidad.




Datos sobre el tecnecio. Obsérvese la forma de representar el estado líquido de dicho elemento.

```
10 REM *****************************
20 REM ***        QUIMICA        ***
30 REM *** Un programa realizado ***
40 REM ***          Por          ***
50 REM *** Carlos A. Maria Morin ***
60 REM ***                       ***
70 REM ***     (C) Ediciones     ***
80 REM ***  Siglo Cultural-1987  ***
90 REM *****************************
100 REM
110 SYMBOL AFTER 230
120 SYMBOL 230,0,15,4,4,4,4,4,4
130 SYMBOL 231,0,240,32,32,32,32,32,32
140 SYMBOL 232,4,8,16,63,111,215,255,127
150 SYMBOL 233,32,16,8,252,254,255,255,254
160 REM
170 REM ****** ACIDO BASICO Y LIQUIDO ******
180 REM
190 SYMBOL 234,0,0,0,0,128,128,170,213
200 SYMBOL 235,0,0,0,0,2,2,170,86
210 SYMBOL 236,0,0,0,0,128,128,128,255
220 SYMBOL 237,0,0,0,0,2,2,2,254
230 SYMBOL 238,128,128,128,128,128,128,128,255
240 SYMBOL 239,2,2,2,2,2,2,2,254
250 REM
260 REM ********** GAS Y SOLIDO ***********
270 REM
280 SYMBOL 240,0,0,0,0,255,138,160,137
290 SYMBOL 241,0,0,0,0,254,18,70,146
300 SYMBOL 242,146,169,130,168,129,164,129,255
310 SYMBOL 243,10,162,10,162,18,70,146,254
320 SYMBOL 245,4,10,10,4,0,0,0,0
330 REM
340 REM *********************************************************
350 REM ************* DIBUJO DE LA TABLA PERIODICA 1 *************
360 REM *********************************************************
370 REM
380 MODE 2
390 BORDER 0
400 INK 0,0:INK 1,0
410 k=1
```

```
420 RESTORE 5530
430 PRINT CHR$(23);CHR$(1)
440 TAG
450 MOVE 184,381
460 PRINT"________________________________________";
470 TAGOFF
480 LOCATE 24,2
490 PRINT"SISTEMA PERIODICO DE LOS ELEMENTOS"
500 PRINT CHR$(23);CHR$(0)
510 LOCATE 14,5
520 PRINT"H                                                  He"
530 LOCATE 14,8
540 PRINT"Li Be                                B  C  N  O  F  Ne"
550 LOCATE 14,11
560 PRINT"Na Mg                                Al Si P  S  Cl Ar"
570 LOCATE 14,14
580 PRINT"K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr"
590 LOCATE 14,17
600 PRINT"Rb Sr Y  Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I  Xe"
610 LOCATE 14,20
620 PRINT"Cs Ba La Hf Ta W  Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn"
630 LOCATE 14,23
640 PRINT"Fr Ra Ac Ku Ha"
650 LOCATE 14,4
660 PRINT"1                                                  2 "
670 LOCATE 14,7
680 PRINT"3  4                                 5  6  7  8  9  10"
690 LOCATE 14,10
700 PRINT"11 12                                13 14 15 16 17 18"
710 LOCATE 14,13
720 PRINT"19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36"
730 LOCATE 14,16
740 PRINT"37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54"
750 LOCATE 14,19
760 PRINT"55 56 57 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86"
770 LOCATE 14,22
780 PRINT"87 88 89 C4 C5"
790 FOR a=1 TO 8
800 READ alto
810 ORIGIN 100,alto
820 IF a=1 THEN DRAW 118,0
830 IF a>=2 AND a<=5 THEN DRAW 432,0
840 IF a>=6 AND a<=7 THEN DRAW 46,0:MOVE 289,0:DRAW 432,0
850 IF a=8 THEN DRAW 23,0:MOVE 408,0:DRAW 432,0
860 NEXT
870 FOR b=1 TO 19
880 READ derecha
890 ORIGIN derecha,30
900 IF b<=2 THEN DRAW 0,335
910 IF b=3 THEN DRAW 0,288
920 IF b>=4 AND b<=6 THEN DRAW 0,190
930 IF b>=7 AND b<=12 THEN MOVE 0,50:DRAW 0,190
940 IF b>=13 AND b<=17 THEN MOVE 0,50:DRAW 0,288
950 IF b>=18 AND b<=19 THEN MOVE 0,50:DRAW 0,335
960 NEXT:GOTO 1470
970 REM
980 REM ***********************************************************
990 REM ************* DIBUJO DE LA TABLA PERIODICA 2 *************
1000 REM ***********************************************************
1010 REM
1020 MODE 2
1030 BORDER 0
1040 INK 1,0:INK 5,3
1050 k=2
1060 RESTORE 5540
1070 PRINT CHR$(23);CHR$(1)
1080 TAG
1090 MOVE 184,381
```

```
1100 PRINT"_____________________________________";
1110 TAGOFF
1120 LOCATE 24,2
1130 PRINT"SISTEMA PERIODICO DE LOS ELEMENTOS"
1140 PRINT CHR$(23);CHR$(0)
1150 LOCATE 20,12
1160 PRINT"57 58 59 60 61 62 63 64 65 66 67 68 69 70 71"
1170 LOCATE 20,15
1180 PRINT"89 90 91 92 93 94 95 96 97 98 99 C0 C1 C2 C3"
1190 LOCATE 20,13
1200 PRINT"La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu"
1210 LOCATE 20,16
1220 PRINT"Ac Th Pa U  Np Pu Am Cm Bk Cf Es Fm Md No Lw"
1230 FOR a=1 TO 3
1240 READ alto
1250 ORIGIN 123,alto
1260 DRAW 384,0
1270 NEXT
1280 FOR b=1 TO 17
1290 READ derecha
1300 ORIGIN derecha,143
1310 DRAW 0,96
1320 NEXT
1330 ORIGIN 0,0
1340 MOVE 128,210
1350 FILL 13
1360 PEN 5
1370 LOCATE 17,13
1380 PRINT"LA"
1390 MOVE 128,160
1400 FILL 13
1410 PEN 5
1420 LOCATE 17,16
1430 PRINT"AC"
1440 PEN 1
1450 GOTO 1950
1460 REM
1470 REM ***********************************************************
1480 REM ***************** MOVIMIENTO DEL CURSOR 1 ***************
1490 REM ***********************************************************
1500 REM
1510 INK 1,26
1520 ORIGIN 0,0
1530 x%=103
1540 y%=335
1550 PRINT CHR$(23);CHR$(1)
1560 GOSUB 1680
1570 RESTORE 5560
1580 WHILE INKEY(18)=-1
1590 IF INKEY(8)=0 THEN GOSUB 1850:x%=MAX(127,x%)-24:GOSUB 1680
1600 IF INKEY(1)=0 THEN GOSUB 1850:x%=MIN(487,x%)+24:GOSUB 1680
1610 IF INKEY(0)=0 THEN GOSUB 1850:y%=MIN(287,y%)+48:GOSUB 1680
1620 IF INKEY(2)=0 THEN GOSUB 1850:y%=MAX(95,y%)-48:GOSUB 1680
1630 WEND
1640 SOUND 1,60,15,15
1650 TAGOFF
1660 GOTO 2260
1670 PRINT CHR$(23);CHR$(0);
1680 IF YPOS=63 THEN x%=MIN(199,x%)
1690 IF YPOS=255 AND XPOS=143 THEN x%=MIN(128,x%)
1700 IF YPOS=255 AND XPOS=144 THEN x%=MAX(391,x%)
1710 IF YPOS=255 AND XPOS=407 THEN x%=MAX(390,x%)
1720 IF YPOS=255 AND XPOS=406 THEN x%=MIN(127,x%)
1730 IF YPOS=303 AND XPOS=143 THEN x%=MIN(128,x%)
1740 IF YPOS=303 AND XPOS=144 THEN x%=MAX(391,x%)
1750 IF YPOS=303 AND XPOS=407 THEN x%=MAX(390,x%)
1760 IF YPOS=303 AND XPOS=406 THEN x%=MIN(127,x%)
1770 IF YPOS=351 AND XPOS=527 THEN x%=MAX(510,x%)
```

```
1780 IF YPOS=351 AND XPOS=526 THEN x%=MIN(103,x%)
1790 IF YPOS=351 AND XPOS=119 THEN x%=MIN(104,x%)
1800 IF YPOS=351 AND XPOS=120 THEN x%=MAX(511,x%)
1810 IF XPOS=143 AND YPOS=303 THEN y%=MIN(287,y%)
1820 IF XPOS>=407 AND XPOS<=503 AND YPOS=303 THEN y%=MIN(287,y%)
1830 IF XPOS>=167 AND XPOS<=383 AND YPOS=207 THEN y%=MIN(191,y%)
1840 IF XPOS>=239 AND XPOS<=527 AND YPOS=111 THEN y%=MAX(95,y%)
1850 FOR ret=1 TO 10
1860 NEXT
1870 TAG
1880 FRAME
1890 MOVE x%,y%
1900 PRINT CHR$(143)+CHR$(143);
1910 MOVE x%,y%+16
1920 PRINT CHR$(143)+CHR$(143);
1930 RETURN
1940 REM
1950 REM ***********************************************************
1960 REM **************** MOVIMIENTO DEL CURSOR 2 *****************
1970 REM ***********************************************************
1980 REM
1990 INK 1,26
2000 ORIGIN 0,0
2010 x%=127
2020 y%=206
2030 PRINT CHR$(23);CHR$(1)
2040 GOSUB 1680
2050 RESTORE 5570
2060 WHILE INKEY(18)=-1
2070 IF INKEY(8)=0 THEN GOSUB 2160:x%=MAX(151,x%)-24:GOSUB 2160
2080 IF INKEY(1)=0 THEN GOSUB 2160:x%=MIN(463,x%)+24:GOSUB 2160
2090 IF INKEY(0)=0 THEN GOSUB 2160:y%=MIN(158,y%)+48:GOSUB 2160
2100 IF INKEY(2)=0 THEN GOSUB 2160:y%=MAX(206,y%)-48:GOSUB 2160
2110 WEND
2120 SOUND 1,60,15,15
2130 TAGOFF
2140 GOTO 2400
2150 PRINT CHR$(23);CHR$(0)
2160 FOR ret=1 TO 10
2170 NEXT
2180 TAG
2190 FRAME
2200 MOVE x%,y%
2210 PRINT CHR$(143)+CHR$(143);
2220 MOVE x%,y%+16
2230 PRINT CHR$(143)+CHR$(143);
2240 RETURN
2250 REM
2260 REM *********************************************************************
2270 REM ********* RECONOCE ELEMENTO EN LA POSICION DEL CURSOR TABLA 1 ********
2280 REM *********************************************************************
2290 REM
2300 RESTORE 5560
2310 FOR ies=335 TO 47 STEP -48
2320 FOR equis=103 TO 511 STEP 24
2330 READ elemento$(1),nombre$,numeroatom$,ebullicion$,fusion$
2340 READ densid$,pesatom$,valen$,estruc$,estado$,acido$,base$,cristal$
2350 IF y%=ies AND x%=equis THEN IF elemento$(1)="La" OR elemento$(1)="Ac" THEN
GOTO 980 ELSE GOTO 2520
2360 NEXT equis,ies
2370 GOTO 1570
2380 REM
2390 REM *********************************************************************
2400 REM ********* RECONOCE ELEMENTO EN LA POSICION DEL CURSOR TABLA 2 ********
2410 REM *********************************************************************
2420 REM
2430 RESTORE 5930
2440 FOR ies=206 TO 158 STEP -48
2450 FOR equis=103 TO 487 STEP 24
```

```
2460 READ elemento$(2),nombre$,numeroatom$,ebullicion$,fusion$
2470 READ densid$,pesatom$,valen$,estruc$,estado$,acido$,base$,cristal$
2480 IF y%=ies AND x%=equis THEN IF (y%=206 AND x%=127) OR (y%=158 AND x%=127) T
HEN GOTO 380 ELSE GOTO 2520
2490 NEXT equis,ies
2500 GOTO 2050
2510 REM
2520 REM ************************************************************
2530 REM ********************* IMPRIME DATOS ***********************
2540 REM ************************************************************
2550 REM
2560 MODE 1
2570 PRINT CHR$(23);CHR$(0)
2580 INK 1,0
2590 TAGOFF
2600 LOCATE 30,3
2610 PRINT "ESTRUCTURA";
2620 LOCATE 30,4
2630 PRINT "CRISTALINA"
2640 IF cristal$="1" THEN GOSUB 4100
2650 IF cristal$="2" THEN GOSUB 4270
2660 IF cristal$="3" THEN GOSUB 4430
2670 IF cristal$="4" THEN GOSUB 4610
2680 IF cristal$="5" THEN GOSUB 4830
2690 IF cristal$="6" THEN GOSUB 5100
2700 IF cristal$="7" THEN GOSUB 5210
2710 IF cristal$="8" THEN GOSUB 5310
2720 IF cristal$="9" THEN GOSUB 5430
2730 LOCATE 31,11:PRINT"ACI./BAS."
2740 IF acido$="1" THEN GOSUB 3770
2750 IF base$="1" THEN GOSUB 3820
2760 LOCATE 34,15:PRINT"ESTADO"
2770 IF estado$="1" THEN GOSUB 3870
2780 IF estado$="2" THEN GOSUB 3920
2790 IF estado$="3" THEN GOSUB 3970
2800 IF estado$="4" THEN GOSUB 4000
2810 dis=LEN(numeroatom$)
2820 LOCATE 14,6
2830 PRINT"NUMERO ATOMICO"
2840 LOCATE 20-INT(dis/2),7
2850 PRINT numeroatom$
2860 LOCATE 1,3
2870 PRINT"PESO";
2880 LOCATE 1,4
2890 PRINT"ATOMICO";
2900 LOCATE 2,5
2910 PRINT pesatom$
2920 LOCATE 1,7
2930 PPINT "PUNTO DE";
2940 LOCATE 1,8
2950 PRINT"EBULLICION";
2960 LOCATE 2,9
2970 PRINT ebullicion$;" "CHR$(245)"C"
2980 LOCATE 1,11
2990 PRINT "PUNTO DE";
3000 LOCATE 1,12
3010 PRINT"FUSION";
3020 LOCATE 2,13
3030 PRINT fusion$;" "CHR$(245)"C"
3040 LOCATE 1,15
3050 PRINT "DENSIDAD";
3060 LOCATE 2,16
3070 PRINT densid$;" g/ml"
3080 dis=LEN(valen$)
3090 LOCATE 17,18
3100 PRINT"VALENCIA";
3110 LOCATE 13,19
3120 PRINT"(La mas estable)";
```

```
3130 LOCATE 20-INT(dis/2),20
3140 PRINT valen$
3150 dis=LEN(estruc$)
3160 LOCATE 10,24
3170 PRINT"ESTRUCTURA ELECTRONICA";
3180 LOCATE 20-INT(dis/2),25
3190 PRINT estruc$
3200 REM
3210 REM ********************************************************
3220 REM ************* IMPRIME CARACTERES GIGANTES ****************
3230 REM ********************************************************
3240 REM
3250 ORIGIN 226,163
3260 DRAW 179,0
3270 MOVE 179,0
3280 DRAW 179,104
3290 MOVE 179,104
3300 DRAW 0,104
3310 DRAW 0,0
3320 ORIGIN 230,167
3330 DRAW 171,0
3340 MOVE 171,0
3350 DRAW 171,98
3360 MOVE 171,96
3370 DRAW 0,96
3380 DRAW 0,0
3390 MOVE -2,-2
3400 FILL 1
3410 ORIGIN 0,0
3420 x=26
3430 y=1
3440 zz=100
3450 mm=240
3460 ll=16*(x-1)
3470 jj=399-(y-1)*16
3480 a$=elemento$(k)
3490 GOSUB 3560
3500 zz=zz+64
3510 INK 1,26
3520 IF INKEY(47)<>0 THEN 3520
3530 SOUND 1,100,15,15
3540 IF k=1 THEN 380
3550 IF k=2 THEN 980
3560 hh=zz
3570 oo=mm
3580 LOCATE x,y
3590 PRINT a$
3600 LOCATE x-12,y
3610 PRINT nombre$
3620 FOR ss=jj TO jj-16 STEP -2
3630 FOR vv=ll TO ll+32 STEP 2
3640 IF TEST(vv,ss) THEN GOSUB 3710
3650 hh=hh+8
3660 NEXT
3670 oo=oo-8
3680 hh=zz
3690 NEXT
3700 RETURN
3710 IF LEN(a$)=1 THEN uno=185 ELSE uno=161
3720 FOR sal=0 TO 7 STEP 2
3730 MOVE hh+uno,oo-sal
3740 DRAWR 6,0
3750 NEXT
3760 RETURN
3770 LOCATE 31,12
3780 PRINT CHR$(234)+CHR$(235);
3790 LOCATE 31,13
3800 PRINT CHR$(238)+CHR$(239)
```

```
3810 RETURN
3820 LOCATE 37,12
3830 PRINT CHR$(236)+CHR$(237);
3840 LOCATE 37,13
3850 PRINT CHR$(238)+CHR$(239)
3860 RETURN
3870 LOCATE 36,16
3880 PRINT CHR$(240)+CHR$(241);
3890 LOCATE 36,17
3900 PRINT CHR$(242)+CHR$(243)
3910 RETURN
3920 LOCATE 36,16
3930 PRINT CHR$(234)+CHR$(235);
3940 LOCATE 36,17
3950 PRINT CHR$(238)+CHR$(239)
3960 RETURN
3970 LOCATE 36,16
3980 PRINT CHR$(143)+CHR$(143)
3990 RETURN
4000 LOCATE 36,16
4010 PRINT CHR$(230)+CHR$(231);
4020 LOCATE 36,17
4030 PRINT CHR$(232)+CHR$(233)
4040 RETURN
4050 REM
4060 REM **********************************************************
4070 REM ********* DIBUJO DE LAS ESTRUCTURAS CRISTALINAS **********
4080 REM **********************************************************
4090 REM
4100 REM +++++++++++++++++ TETRAEDRO ++++++++++++++++++++
4110 REM
4120 ORIGIN 525,250
4130 DRAW 25,0:DRAW 25,48
4140 DRAW 25,48:DRAW 0,48
4150 DRAW 0,0:DRAW 15,14
4160 MOVE 25,0:DRAW 38,14
4170 MOVE 0,48:DRAW 15,62
4180 MOVE 24,48:DRAW 38,62
4190 ORIGIN 538,265
4200 DRAW 25,0:MOVE 25,0
4210 DRAW 25,48:MOVE 25,48
4220 MOVE 25,48:DRAW 25,48
4230 MOVE 25,48:DRAW 0,48
4240 DRAW 0,0
4250 RETURN
4260 REM
4270 REM +++++++++++++++++ TETRAGONAL +++++++++++++++++++
4280 REM
4290 ORIGIN 525,250
4300 DRAW 40,0:DRAW 40,48
4310 DRAW 40,48:DRAW 0,48
4320 DRAW 0,0:DRAW 15,14
4330 MOVE 40,0:DRAW 55,14
4340 MOVE 0,48:DRAW 15,62
4350 MOVE 40,48:DRAW 55,62
4360 ORIGIN 538,265
4370 DRAW 40,0:MOVE 40,0
4380 DRAW 40,48:MOVE 40,48
4390 DRAW 40,48:MOVE 40,48
4400 DRAW 0,48:DRAW 0,0
4410 RETURN
4420 REM
4430 REM ++++++++++++++++ CUBICA SIMPLE ++++++++++++++++++
4440 REM
4450 ORIGIN 525,250
4460 DRAW 50,0:MOVE 50,0
4470 DRAW 50,48:MOVE 50,48
4480 DRAW 50,48:MOVE 50,48
```

```
4490 DRAW 0,48:DRAW 0,0
4500 DRAW 15,14:MOVE 50,0
4510 DRAW 65,14:MOVE 0,48
4520 DRAW 15,62:MOVE 50,48
4530 DRAW 65,62
4540 ORIGIN 538,265
4550 DRAW 50,0:MOVE 50,0
4560 DRAW 50,48:MOVE 50,48
4570 DRAW 50,48:MOVE 50,48
4580 DRAW 0,48:DRAW 0,0
4590 RETURN
4600 REM
4610 REM +++++++++ CUBICA CON CENTRO EN UNA CARA ++++++++++
4620 REM
4630 ORIGIN 525,250
4640 DRAW 50,0:MOVE 50,0
4650 DRAW 50,48:MOVE 50,48
4660 DRAW 50,48:MOVE 50,48
4670 DRAW 0,48:DRAW 0,0
4680 DRAW 15,14:MOVE 50,0
4690 DRAW 65,14:MOVE 0,48
4700 DRAW 15,62:MOVE 50,48
4710 DRAW 65,62
4720 FOR l=1 TO 4
4730 FOR o=1 TO 4
4740 PLOT 24+l,23+o
4750 NEXT o,l
4760 ORIGIN 538,265
4770 DRAW 50,0:MOVE 50,0
4780 DRAW 50,48:MOVE 50,48
4790 DRAW 50,48:MOVE 50,48
4800 DRAW 0,48:DRAW 0,0
4810 RETURN
4820 REM
4830 REM ++++++++ CUBICA CON CENTRO EN EL INTERIOR ++++++++
4840 REM
4850 ORIGIN 525,250
4860 DRAW 50,0:MOVE 50,0
4870 DRAW 50,48:MOVE 50,48
4880 MOVE 50,48:DRAW 50,48
4890 MOVE 50,48:DRAW 0,48
4900 DRAW 0,0
4910 DRAW 15,14:MOVE 50,0
4920 DRAW 65,14:MOVE 0,48
4930 DRAW 15,62:MOVE 50,48
4940 DRAW 65,62
4950 FOR l=1 TO 8
4960 FOR o=1 TO 8
4970 PLOT 29+l,27+o
4980 NEXT o,l
4990 MOVE 15,14:DRAW 50,48
5000 MOVE 0,0: DRAW 65,62
5010 MOVE 50,0:DRAW 15,62
5020 MOVE 65,14:DRAW 0,48
5030 ORIGIN 538,265
5040 DRAW 50,0:MOVE 50,0
5050 DRAW 50,48:MOVE 50,48
5060 DRAW 50,48:MOVE 50,48
5070 DRAW 0,48:DRAW 0,0
5080 RETURN
5090 REM
5100 REM ++++++++++++++++++++ ROMBO ++++++++++++++++++++
5110 REM
5120 ORIGIN 535,255
5130 DRAW 45,0:MOVE 45,0
5140 DRAW 7,20:DRAW 0,0
5150 MOVE 45,0:DRAW 30,40
5160 DRAW 7,20:MOVE 0,0
```

```
5170 DRAW -15,40:DRAW 7,20
5180 MOVE 28,40:DRAW -15,40
5190 RETURN
5200 REM
5210 REM ++++++++++++++++++ ROMBOEDRICA +++++++++++++++++++
5220 REM
5230 ORIGIN 525,270
5240 DRAW 50,0:DRAW 25,40
5250 DRAW 0,0:DRAW 0,28
5260 DRAW 25,40:DRAW 50,29
5270 DRAW 50,0:MOVE 0,0
5280 DRAW 25,-12:DRAW 50,0
5290 RETURN
5300 REM
5310 REM ++++++++++++++++++ MONOCICLICA +++++++++++++++++++
5320 REM
5330 ORIGIN 525,250
5340 DRAW 40,0:MOVE 0,0
5350 DRAW -10,35:MOVE 40,0
5360 DRAW 50,35:MOVE 40,0
5370 DRAW 50,35:MOVE -10,35
5380 DRAW 0,70:MOVE 0,70
5390 DRAW 40,70:MOVE 50,35
5400 DRAW 40,70
5410 RETURN
5420 REM
5430 REM ++++++++++++++++++++ EXAGONAL ++++++++++++++++++++
5440 REM
5450 ORIGIN 550,250
5460 DRAW 0,50:DRAW 25,37
5470 DRAW 25,12:DRAW 0,0
5480 MOVE 0,50:DRAW -25,37
5490 DRAW -25,12:DRAW 0,0
5500 MOVE -25,12:DRAW 25,37
5510 MOVE 25,12:DRAW -25,37
5520 RETURN
5530 DATA 30,78,127,175,223,271,318,364,99,123,147,171,195,219,243,267,291,315,3
40,364,388,412,436,460,484,508,532
5540 DATA 143,191,240,123,147,171,195,219,243,267,291,315,339,363,387,411,435,45
9,483,507
5550 REM
5560 REM **********************************************************************
5570 REM **************************** DATAS ***********************************
5580 REM **********************************************************************
5590 REM
5600 DATA "H","HIDROGENO",1,-252.7,-259.2,0.071,1.0079,1,"1s1",1,1,1,9, , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
 , , , , , , , , , , , , , , , , , , , , ,
5610 DATA , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, ,
5620 DATA , , , , , , , , , , , ,
5630 DATA "He","HELIO",2,-268.9,-269.7,0.126,4.0026,----,"1s2",1,0,0,9,"Li","LIT
IO",3,1330,180.5,0.53,6.939,1,"1s2 2s1",3,0,1,5,"Be"
,"BERILIO",4,27770,1277,1.85,9.0122,2,"1s2 2s2",3,1,1,9
5640 DATA , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , ,
5650 DATA , , , , ,
5660 DATA "B","BORO",5,----,(2030),2.34,10.811,3,"1s2 2s2 2p1",3,1,0,9,"C","CARB
ONO",6,4830,3727,2.26,12.011,+-4  2,"1s2 2s2 2p2",3,
1,0,9,"N","NITROGENO",7,-195.8,-210,.81,14.006,+-3  5  4  2,"1s2 2s2 2p3",1,1,0,
9
5670 DATA "O","OXIGENO",8,-183,-218.8,1.14,15.999,-2,"1s2 2s2 2p4",1,0,0,3,"F","
FLUOR",9,-188.2,-219.6,1.505,18.998,-1,"1s2 2s2 2p5"
,1,0,0,0,"Ne","NEON",10,-246,-248.6,1.20,20.183,----,"1s2 2s2 2p6",1,0,0,4
```

```
5680 DATA "Na","SODIO",11,892,97.8,0.97,22.989,1,"Ne 3s1",3,0,1,5,"Mg","MAGNESIO
",12,1107,650,1.74,24.312,2,"Ne 3s2",3,0,1,9, , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , ,
5690 DATA , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , ,
5700 DATA "Al","ALUMINIO",13,2450,660,2.70,26.981,3,"Ne 3s2 3p1",3,1,1,4,"Si","S
ILICIO",14,2680,1410,2.33,28.086,4,"Ne 3s2 3p2",3,1,
1,6,"P","FOSFORO",15,280 b,44.2 b,1.82 b,30.973,+-3  5  4,"Ne 3s2 3p2",3,1,0,3
5710 DATA "S","AZUFRE",16,444.6,119.0,2.07,32.064,+-2  4  6,"Ne 3s2 3p4",3,1,0,0
,"Cl","CLORO",17,-34.7,-101.0,1.56,35.453,+-1  3  5
 7,"Ne 3s2 3p5",1,1,0,1,"Ar","ARGON",18,-185.8,-189.4,1.40,39.948,----,"Ne 3s2 3
p6",1,0,0,4
5720 DATA "K","POTASIO",19,760,63.7,0.86,39.102,1,"Ar 4s1",3,0,1,5,"Ca","CALCIO"
,20,1440,383,1.55,40.08,2,"Ar 4s2",3,0,1,4,"Sc","ESC
ANDIO",21,2730,1539,3.0,44.956,3,"Ar 3d1 4s2",3,0,1,9
5730 DATA "Ti","TITANIO",22,3260,1668,4.51,47.90,4  3,"Ar 3d2 4s 2",3,1,1,9,"V",
"VANADIO",23,3450,1900,6.1,50.942,5  4  3  2,"Ar 3d3
 4s2",3,1,1,5,"Cr","CROMO",24,2665,1875,7.19,51.996,6  3  2,"Ar 3d5 4s1",3,1,0,5

5740 DATA "Mn","MANGANESO",25,2150,1245,7.43,54.938,7  6  4  2  3,"Ar 3d5 4s2",3
,1,0,3
5750 DATA "Fe","HIERRO",26,3000,1536,7.86,55.847,2  3,"Ar 3d6 4s2",3,1,1,5,"Co",
"COBALTO",27,2900,1495,8.9,58.933,2  3,"Ar 3d7 4s2",
3,1,1,9,"Ni","NIQUEL",28,2730,1453,8.9,58.71,2  3,"Ar 3d8 4s2",3,0,1,4
5760 DATA "Cu","COBRE",29,2595,1083,8.96,63.54,2  1,"Ar 3d10 4s1",3,0,1,4,"Zn","
CINC",30,906,419.5,7.14,65.37,2,"Ar 3d10 4s2",3,1,1,
9,"Ga","GALIO",31,2237,29.8,5.91,69.72,3,"Ar 3d10 4s2 4p1",2,1,1,2
5770 DATA "Ge","GERMANIO",32,2830,937.4,5.32,72.59,4,"Ar 3d10 4s2 4p2",3,1,1,6,"
As","ARSENICO",33,613,817,5.72,74.922,+-3  5,"Ar 3d1
0 4s2 4p3",3,1,0,7,"Se","SELENIO",34,685,217,4.79,78.96,-2  4  6,"Ar 3d10 4s2 4p
4",3,1,0,9
5780 DATA "Br","BROMO",35,58,-7.2,3.12,79.909,+-1  5","Ar 3d10 4s2 4p5",2,1,0,2,
"Kr","KRIPTON",36,-152,-157.3,2.6,83.80,----,"Ar 3d1
0 4s2 4p6",1,0,0,4,"Rb","RUBIDIO",37,688,38.9,1.53,85.47,1,"Kr 5s1",3,0,1,5

5790 DATA "Sr","ESTRONCIO",38,1380,768,2.6,87.62,2,"Kr 5s2",3,0,1,4,"Y","ITRIO",
39,2927,1509,4.47,88.905,3,"Kr 4d1 5s2",3,0,1,9,"Zr"
,"CIRCONIO",40,3580,1852,6.49,91.22,4,"Kr 4d4 5s1",3,1,1,9
5800 DATA "Nb","NIOBIO",41,3300,2468,8.4,92.906,5  3,"Kr 4d4 5s1",3,1,0,5,"Mo","
MOLIBDENO",42,5560,2610,10.2,95.94,6  5  4  3  2,"Kr
 4d5 5s1",3,1,0,5,"Tc","TECNECIO",43,----,2140,11.5,(98),7,"Kr 4d5 5s2",4,1,0,0
5810 DATA "Ru","RUTENIO",44,4900,2500,12.2,101.07,2  3  4  6  8,"Kr 4d7 5s1",3,1
,0,9,"Rh","RODIO",45,4500,1966,12.4,102.90,2  3  4,"
Kr 4d8 5s1",3,1,0,4,"Pd","PALADIO",46,3980,1552,12.0,106.4,2  4,"Kr 4d10 5s0",3,
0,1,4
5820 DATA "Ag","PLATA",47,2210,960.8,10.5,107.87,1,"Kr 4d10 5s1",3,1,1,4,"Cd","C
ADMIO",48,765,320.9,8.65,112.40,2,"Kr 4d10 5s2",3,0,
1,9,"In","INDIO",49,2000,156.2,7.31,114.82,3,"Kr 4d10 5s2 5p1",3,1,1,1
5830 DATA "Sn","ESTA\O",50,2270,231.9,7.30,118.69,4  2,"Kr 4d10 5s2 5p2",3,1,1,1
,"Sb","ANTIMONIO",51,1380,630.5,6.62,121.75,+-3  5,"
Kr 4d10 5s2 5p3",3,1,0,7,"Te","TELURIO",52,989.8,449.5,6.24,127.60,-2  4  6,"Kr
4d10 5s2 5p4",3,1,0,9
5840 DATA "I","YODO",53,183,113.7,4.94,128.90,+-1  5  7,"Kr 4d10 5s2 5p5",3,1,0,
2,"Xe","XENON",54,-108,-111.9,3.06,131.30,----,"Kr 4
d10 5s2 5p6",1,0,0,4,"Cs","CESIO",55,690,28.7,1.90,132.90,1,"Xe 6s1",2,0,1,5
5850 DATA "Ba","BARIO",56,1640,714,3.5,137.34,2,"Xe 6s2",3,0,1,5,"La", , , , , ,
 , , , , , , , ,"Hf","HAFNIO",72,5400,2222,13.1,178.49
,4,"Xe 4f14 5d2 6s2",3,1,0,9
5860 DATA "Ta","TANTALO",73,5425,2996,16.6,180.94,5,"Xe 4f14 5d3 6s2",3,1,0,5,"W
","VOLFRAMIO",74,5930,3410,19.3,183.85,6  5  4  3  2
,"Xe 4f14 5d4 6s2",3,1,0,5,"Re","RENIO",75,5900,3180,21.0,186.2,7  6  4  2  -1,"
Xe 4f14 5d5 6s2",3,1,0,9
5870 DATA "Os","OSMIO",76,5500,3000,22.6,190.2,2  3  4  6  8,"Xe 4f14 5d6 6s2",3
,1,0,9,"Ir","IRIDIO",77,5300,2454,22.5,192.2,2  3  4
  6,"Xe 4f14 5d7 6s2",3,0,1,4,"Pt","PLATINO",78,4530,1769,21.4,195.09,2  4,"Xe 4
f14 5d9 6s1",3,0,1,4
```

```
5880 DATA "Au","ORO",79,2970,1063,19.3,196.96,3  1,"Xe 4f14 5d10 6s1",3,1,1,4,"H
g","MERCURIO",80,357,-38.4,13.6,200.59,2  1,"Xe 4f14
 5d10 6s2",2,0,1,7,"Tl","TALIO",81,1457,303,11.85,204.37,3  1,"Xe 4f14 5d10 6s2
6p1",3,0,1,9
5890 DATA "Pb","PLOMO",82,1725,327.4,11.4,207.19,4  2,"Xe 4f14 5d10 6s2 6p2",3,1
,1,4,"Bi","BISMUTO",83,1560,271.3,9.8,208.98,3  5,"X
e 4f14 5d10 6s2 6p3",3,1,0,7,"Po","POLONIO",84,----,254,(9.2),(210),2  4,"Xe 4f1
4 5d10 6s2 6p4",3,1,1,8
5900 DATA "At","ASTATO",85,----,(302),----,(210),+-1  3  5  7,"Xe 4f14 5d10 6s2
3p5",3,0,0,0,"Rn","RADON",86,(-61.8),(-7.1),----,(22
2),----,"Xe 4f14 5d10 6s2 6p6",1,0,0,4,"Fr","FRANCIO",87,----,(27),----,223,3  4
,"Rn 7s1",2,0,1,5
5910 DATA "Ra","RADIO",88,----,700,5.0,226,2,"Rn 7s2",3,0,1,0,"Ac", , , , , , ,
, , , , , ,"Ku","KURCHATOVIO",104, , , ,261, , , , ,
 , ,"HA","HANIO",105, , ,262, , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , ,
5920 DATA , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
, , , , ,
5930 DATA , , ,
5940 DATA , , , , , , , , , , , , , , , , , , , , , , ,"La","LANTANO",57,3470,920,
6.17,138.91,3,"Xe 5d1 6s2",3,0,1,9,"Ce","CERIO",58,3
468,795,6.67,140.12,3  4,"Xe 4f2 5d0 6s2",3,0,1,4,"Pr","PRASEODIMIO",59,3127,935
,6.77,140.90,3  4,"Xe 4f3 5d0 6s2",3,0,1,9
5950 DATA "Nd","NEODIMIO",60,3027,1024,7.00,144.24,3,"Xe 4f4 5d0 6s2",3,0,1,9,"P
m","PROMETIO",61,----,(1027,----,(147),3,"Xe 4f5 d0
6s2",4,0,1,9,"Sm","SAMARIO",62,1900,1072,7.54,150.35,3  2,"Xe 4f6 5d0 6s2",3,0,1
,7
5960 DATA "Eu","EUROPIO",63,1439,826,5.26,151.96,3  2,"Xe 4f7 5d0 6s2",3,0,1,5,"
Gd","GADOLINIO",64,3000,1312,7.89,157.25,3,"Xe 4f7 5
d1 6s2",3,0,1,9,"Tb","TERBIO",65,2800,1356,8.27,158.92,3  4,"Xe 4f8 5d0 6s2",3,0
,1,9
5970 DATA "Dy","DISPROSIO",66,2600,1407,8.54,162.50,3,"Xe 4f10 5d0 6s2",3,0,1,9,
"Ho","HOLMIO",69,2600,1461,8.80,164.93,3,"Xe 4f11 5d
0 6s2",3,0,1,9,"Er","ERBIO",68,2900,1497,9.05,167.26,3,"Xe 4f12 5d0 6s2",3,0,1,9

5980 DATA "Tm","TULIO",69,1727,1545,9.33,168.93,3  2,"Xe 4f13 5d0 6s2",3,0,1,9,"
Yb","YTERBIO",70,1427,824,6.98,173.04,3  2,"Xe 4f14
5d0 6s2",3,0,1,4,"Lu","LUTECIO",71,3327,1652,9.84,174.97,3,"Xe 4f14 5d0 6s2",3,0
,1,9
5990 DATA , , , , , , , , , , , , , , , , , , , , , , , , , , ,"Ac","ACTINIO",89,-
---,1050,----,227,3,"Rn 6d1 7s2",3,0,0,0,"Th","TORIO
",90,3850,1750,11.7,232.03,4,"Rn 5f0 6d2 7s2",3,0,1,4
6000 DATA "Pa","PROTACTINIO",91,----,(1230),15.4,(231),5  4,"Rn 5f2 6d1 7s2",3,0
,1,0,"U","URANIO",92,3818,1132,19.07,238.03,6  5  4
 3,"Rn 5f3 6d1 7s2",3,1,1,2,"Np","NEPTUNIO",93,----,637,19.5,(237),6  5  4  3,"R
n 5f4 6d1 7s2",4,1,1,0
6010 DATA "Pu","PLUTONIO",94,3235,640,----,(242),6  5  4  3,"Rn 5f6 6d0 7s2",4,1
,1,0,"Am","AMERICIO",95,----,----,11.7,(243),6  5  4
  3,"Rn 5f7 6d0 7s2",4,0,0,0,"Cm","CURIO",96,----,----,----,(247),3,"Rn 5f7 6d1
7s2",4,0,0,0
6020 DATA "Bk","BERKELIO",97,----,----,----,(247),4  3,"Rn 5f0 6d0 7s2",4,0,0,0,
"Cf","CALIFORNIO",98,----,----,----,(249),3,"Rn 5f10
 6d0 7s2",4,0,0,0,"Es","EINSTENIO",99,----,----,----,(254),----,"Rn 5f11 6d0 7s2
",4,0,0,0
6030 DATA "Fm","FERMIO",100,----,----,----,(253),----,"Rn 5f12 6d0 7s2",4,0,0,0,
"Md","MENDELEVIO",101,----,----,----,(256),----,"Rn
5f13 6d0 7s2",4,0,0,0,"No","NOBELIO",102,----,----,----,(256),----,"Rn 5f14 6d0
7s2",4,0,0,0
6040 DATA"Lw","LAWRENCIO",103,----,----,----,(257),----,"Rn 5f14 6d1 7s2",4,0,0,
0
```

Como en el dibujo de la tabla periódica no caben las llamadas tierras raras, si queremos ver alguna de ellas, sólo tenemos que colocar el cursor por encima del elemento 57 (lantanio) o del 89 (actinio) para ver la subtabla de los Lantánidos y Actánidos (tierras raras). Una vez en esta nueva tabla, tenemos que mover el cursor al elemento que queramos y pulsar la tecla ENTER para visualizarlo.

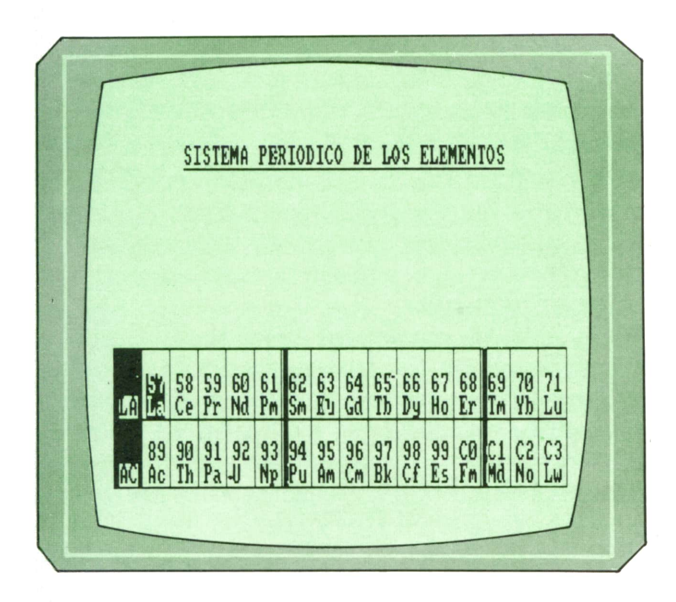
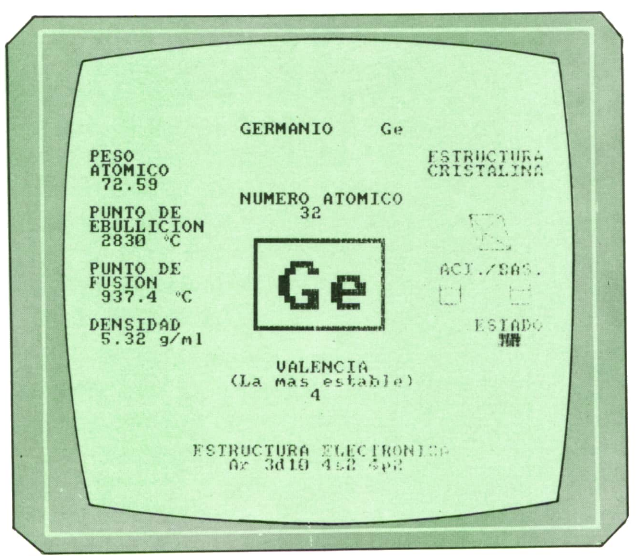
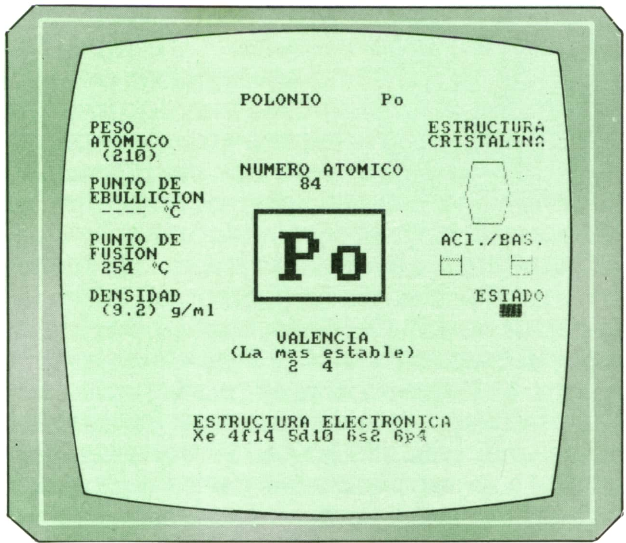



Tabla periódica con las tierras raras.

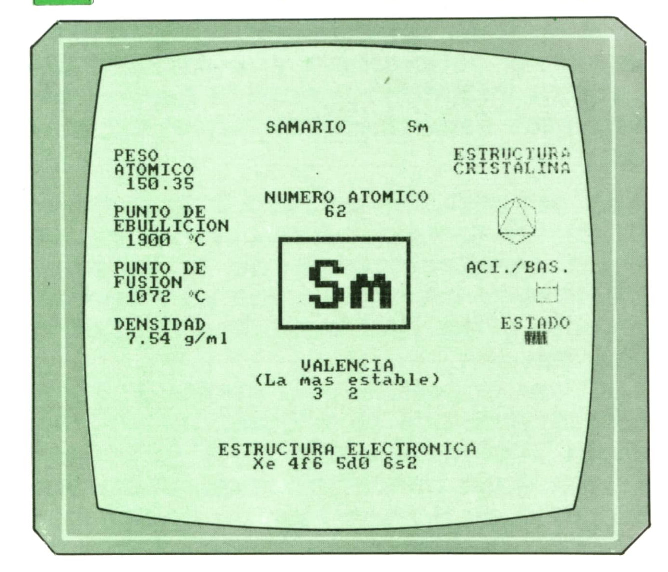
Aunque el programa es algo largo y puede resultar un poco difícil su introducción en el ordenador, los resultados obtenidos merecen todo el trabajo. Hay que decir que se debe tener mucho cuidado a la hora de introducir las líneas DATA para que todos los datos sean correctos.




Información sobre el germanio.



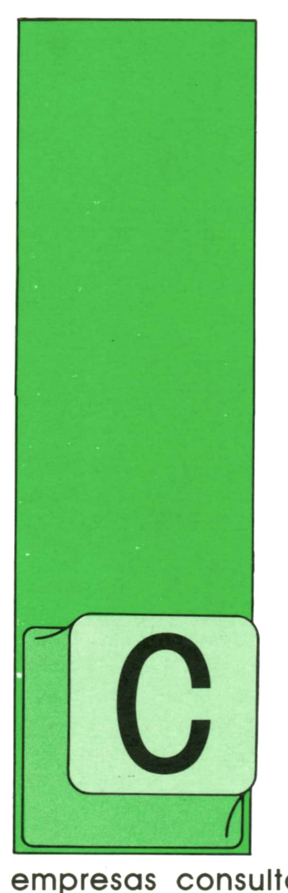

El polonio. Al ser un elemento neutro (ni ácido, ni básico), se dibujan las dos cubetas.

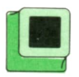


El samario, al ser un elemento sólido y básico, nos aparece con un rectángulo y con una cubeta bajo la palabra «básico».



TECNICAS DE ANALISIS





El método Merise

COMO ya hemos comentado, este método de concepción, diseño y control de la implementación de proyectos informáticos ha sido desarrollado por un grupo de empresas consultoras y fabricantes de software francesas y ha sido aceptado como estándar de la ingeniería informática por la AFNOR (Asociación Francesa para la Normalización). A partir de los primeros documentos preparados y de las experiencias posteriores, los coautores han publicado (1985-86, Editions de l'Organisation) una especie de guía maestra del sistema «La Methode Merise, demarches et pratiques».

Según sus creadores, el método intenta conseguir un conjunto de objetivos en el dominio de aplicación al que va destinado: la concepción y realización de sistemas informáticos para la gestión de empresas. Estos objetivos son los siguientes:

a) ser utilizable en todo tipo de organizaciones tanto públicas como privadas e independientemente de su tamaño, del nivel de informatización ya existente en ellas y del problema que se quiera abordar;

b) ser aplicable con cualquier solución técnica que se adopte: tratamiento de la información por lotes (procesos batch) o en conversacional; utilización de bases de datos o manejo de ficheros clásicos; medios informáticos centralizados o informática distribuida; programación con lenguajes de cuarta generación y otras herramientas disponibles o programación clásica con lenguajes convencionales;

c) prever desde el comienzo la evolución futura de los tratamientos en vez de concebir (como es tan usual desgraciadamente) el sistema en general y las informaciones a manejar en particular, en función de los tratamientos que se observan al comienzo del diseño;

d) evitar en las sucesivas etapas del desarrollo cualquier tipo de vuelta atrás (que suelen ser tan costosas en medios, en eficacia y en tiempo);

e) permitir una utilización parcial de los estudios establecidos, evitando que la solución de los problemas particulares esté hipotecada por los planteamientos generales a nivel organización y que soporte excesivas servidumbres derivadas de la mecanización global;

f) definir un número suficiente de elementos invariantes, a diferentes niveles, para disponer de puntos sólidos de anclaje de todo el edificio organizativo que se está construyendo y evitar así que, a lo largo del desarrollo y evolución de los sistemas, se produzca el «síndrome del castillo de naipes».

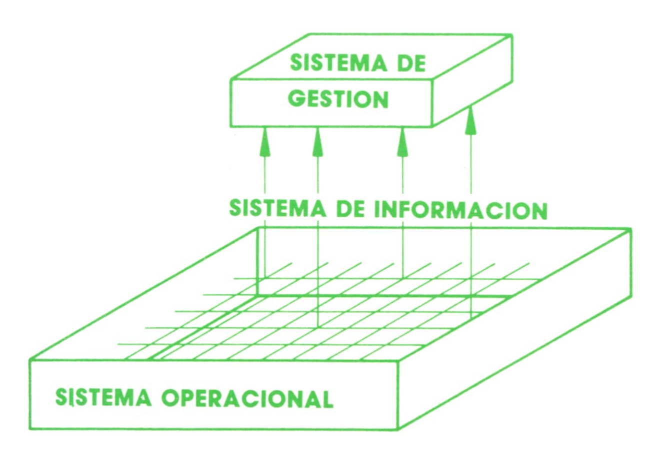
Para asegurar este aspecto último indicado, será necesario definir tres tipos de soluciones antes de abordar cada uno de los elementos que cubran las necesidades de un punto particular. Estas tres soluciones globales son:

— Solución **conceptual** independiente de los papeles respectivos del hombre y de la máquina y del nivel de automatización actual o futuro;

— Solución **lógica** que marca, precisamente los papeles respectivos de hombres y máquinas, su distribución geográfica y el funcionamiento de los canales de distribución de información a todos los niveles. No se consideran en la concepción de esta solución, sin embargo, los medios de tratamiento de la información, ni la situación ni el modo de almacenamiento de los datos actuales o futuros;

— Solución **física** adaptada a un conjunto de medios materiales (equipos) y herramientas software (lenguajes de cuarta generación, gestores de bases de datos, etc.).

El objetivo global consiste en descomponer el sistema de información que se está diseñado y estructurando, identificando sus subconjuntos componentes pero manteniendo entre ellos una serie de fronteras invariantes para asegurar, a todo lo largo del desarrollo del proyecto, la visión de conjunto necesaria para mantener la homogeneidad y coherencia de todos los elementos.




Sistemas estructurales de una organización.

Para ello, es necesario distinguir entre los órganos visibles de la empresa (talleres, medios de producción, centros de trabajo...) y los «sistemas» subyacentes a ellos y que les hacen funcionar. Estos sistemas subyacentes en la organización de la empresa son de tres tipos:

a) **sistemas de gestión,** en los que se toman las decisiones sobre el desarrollo de las tareas de la empresa;
b) **sistemas de información,** que «alimentan» de datos a los anteriores y constituyen, en definitiva, los canales de comunicación en la organización;
c) **sistemas operacionales,** que realizan las tareas cotidianas del funcionamiento de la empresa.

El objeto del método es desarrollar un sistema de información fiable y eficaz. De la calidad de él depende en gran medida la fiabilidad del sistema de gestión e, incluso, la del sistema operacional.

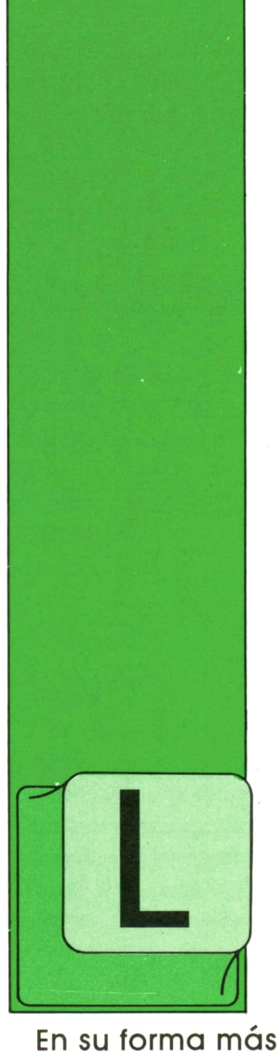
Aun cuando se considera la empresa como un conjunto de dominios diferentes (la producción, las compras, la gestión de personal...) será necesario tener en cuenta en cada uno de estos dominios los tres sistemas considerados: posteriormente se podrán establecer las relaciones entre sistemas.

Una de las ventajas básicas del método es la subdivisión en etapas sucesivas de los desarrollos, con lo cual resulta enormemente flexible y se adapta a diferentes situaciones de las empresas en cuanto a su nivel de mecanización y de organización en general.

TECNICAS DE PROGRAMACION



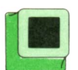
Instrucciones de transferencia



LEGAMOS al fin a la última familia de instrucciones de control que vamos a detallar aquí: la tan denostada y utilizada instruccion de transferencia.

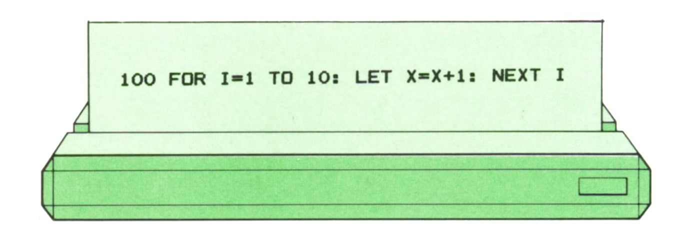
En su forma más sencilla, la instrucción de transferencia permite interrumpir la marcha normal de un programa (que consiste en ejecutar sucesivamente instrucción tras instrucción), especificando el punto exacto donde debe continuar la ejecución. Es decir: cuando un programa llega a una instrucción de transferencia, se produce un salto que puede dar lugar a que la instrucción siguiente se encuentre en cualquier otro punto del programa, más o menos alejada del lugar de origen del salto.

Para conseguirlo, es evidente que hay que tener alguna manera de dar nombre a las instrucciones del programa, pues en caso contrario no podríamos definir fácilmente el sitio donde deseamos que continúe la ejecución. Por ello, antes de hablar con detalle de las instrucciones de transferencia, será preciso explicar los nombres de las instrucciones o «etiquetas».



Etiquetas en BASIC

Cada uno de los lenguajes de programación tiene su propia forma de definir etiquetas, o nombres de las instrucciones. En BASIC, por ejemplo, casi todas las instrucciones tienen etiqueta, que en este caso es, simplemente, un número de instrucción. La única excepción es el bloque secuencial de instrucciones, donde tan sólo la primera instrucción del bloque tiene número de instrucción. Veamos un ejemplo:



En el ejemplo anterior tenemos tres instrucciones que forman un bloque secuencial. Al bloque entero se le asigna una sola etiqueta (el número 100) que, en realidad, corresponde tan sólo a la primera instrucción del bucle. Las instrucciones «LET X=X+1» Y «NEXT 1» no tienen etiqueta y por tanto no son accesibles directamente a través de una instrucción de transferencia.

En BASIC existen varias restricciones a las etiquetas que pueden tener las instrucciones. En primer lugar, cada línea del programa (que puede ser una ins-

trucción única o un bloque secuencial de instrucciones) está obligada a tener número de instrucción y no puede prescindir de él. En segundo lugar, los números de las instrucciones tienen que ser crecientes, aunque no necesariamente consecutivos. Es decir, las siguientes secuencias de etiquetas son válidas en un programa BASIC:

10 20 30 40 50 60 70 80 90 100...
1001 1002 1003 1004 2001 2003 2004...
2 6 10 15 21 22 25 98 101...

mientras que las siguientes son incorrectas:

10 20 30 25 33 40 30 90 100...
1001 1002 1003 1004 201 203 204...
2 6 10 15 21 22 25 25...

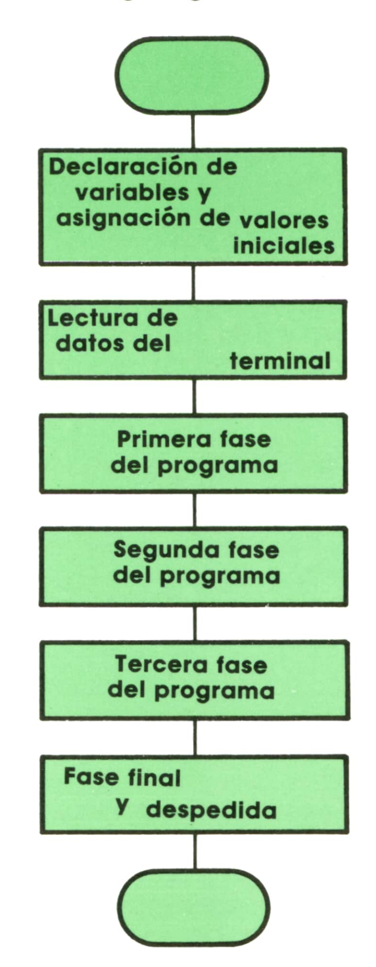
La primera secuencia es incorrecta porque los números de instrucción aparecen en cualquier orden: unas veces crecen y otras disminuyen. La segunda está formada por dos secuencias que en sí son correctas (1001, 1002, 1003, 1004 y 201 203 204), pero la segunda secuencia debería haber sido anterior a la primera. El tercer ejemplo es inválido porque no pueden repetirse los números de las instrucciones.

Es práctica común en BASIC poner a las instrucciones consecutivas números sucesivos que difieran entre sí en diez unidades. Es decir, adoptar la secuencia

10 20 30 40 50 60 70 80 90 100...

Esto tiene la ventaja de que, si es necesario cambiar nuestro programa e introducir nuevas instrucciones entre las anteriormente definidas, podremos utilizar los números intermedios (que permiten insertar hasta nueve instrucciones entre cada dos del programa original) y no nos veremos obligados a cambiar la numeración del programa entero, como ocurriría indefectiblemente si añadiéramos una sola instrucción a un programa cuya secuencia de etiquetas fuera estrictamente consecutiva.

Como se observará, el tipo de etiquetas permitido por el lenguaje BASIC es muy poco satisfactorio. Utilizar números como nombres de las instrucciones no ayuda nada a la comprensión de los programas y hace realmente necesario poner comentarios abundantes que palíen este defecto. Otro método que suele emplearse consiste en utilizar números especiales, fáciles de recordar, al comienzo de las partes más importantes de nuestro programa. Por ejemplo, supongamos que éste puede representarse por medio del siguiente organigrama resumido:



Supongamos que las diversas secciones y fases del programa abarcan las siguientes instrucciones:

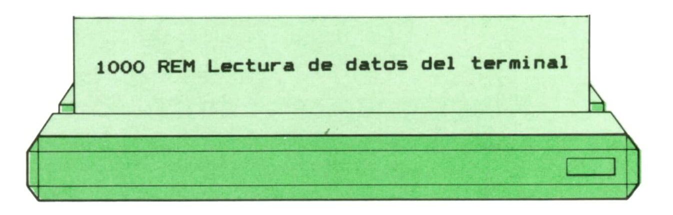
1. Declaración de variables y asignación de valores iniciales: 40 instrucciones.
2. Lectura de datos del terminal: 8 instrucciones.
3. Primera fase: 25 instrucciones.
4. Segunda fase: 14 instrucciones.
5. Tercera fase: 85 instrucciones.
6. Fase final y despedida: 13 instrucciones.

Los números de instrucción recomendables serán los siguientes:

1. Declaración de variables y asignación de valores iniciales: 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160

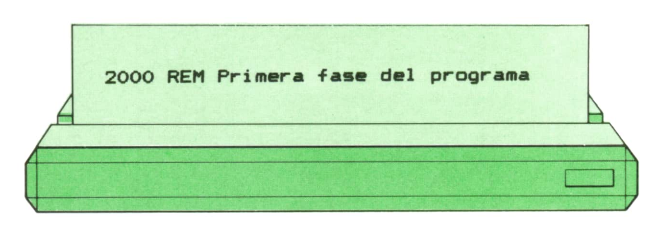
170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400.

2. Lectura de datos del terminal: 1010 1020 1030 1040 1050 1060 1070 1080. Añadir, además, al principio de esta zona, la instrucción siguiente:

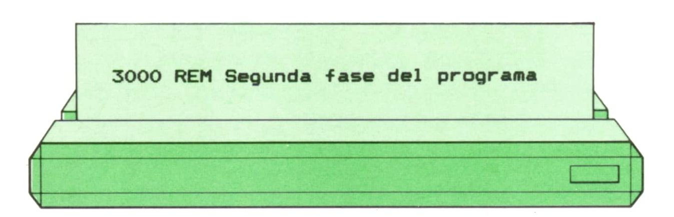


o un comentario parecido.

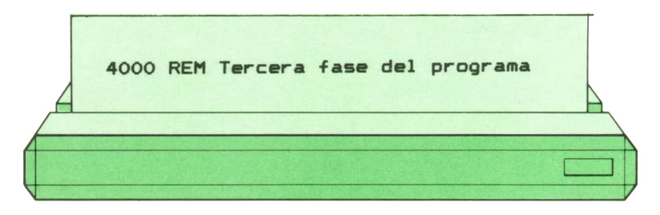
3. Primera fase: 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100 2110 2120 2130 2140 2150 2160 2170 2180 2190 2200 2210 2220 2230 2240 2250. Añadir además, al principio de esta zona, la siguiente instrucción de comentario, o una equivalente:



4. Segunda fase: 3010 3020 3030 3040 3050 3060 3070 3080 3090 3100 3110 3120 3130 3140, además del siguiente comentario, situado al principio de la zona:

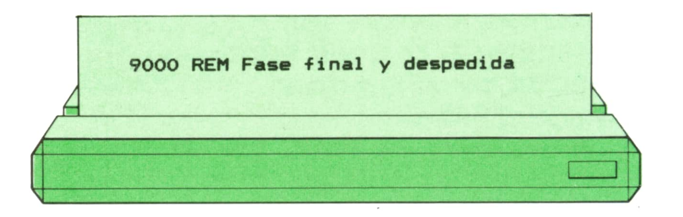


5. Tercera fase: el comentario correspondiente.



seguido de las 85 instrucciones, con los números comprendidos entre 4010 y 4850, de diez en diez.

6. Fase final y despedida: Primero un comentario y después los números 9010 9020 9030 9040 9050 9060 9070 9080 9090 9110 9120 9130.



Obsérvese que hemos utilizado los múltiplos de 1000 al comienzo de cada una de las partes iniciales del programa, pero hemos saltado hasta el 9000 al llegar a la fase final. De esta manera, si algún día tuviésemos que realizar una revisión importante de nuestro programa que nos obligara a introducir una cuarta fase, dispondríamos de los números comprendidos entre 5000 y 8999. Además, la tercera fase es muy larga (casi llega hasta el número 5000), por lo que podría ocurrir que llegáramos a rebasarlo si la corrección del programa obliga a introducir más instrucciones. Por tanto, será conveniente en cualquier caso dejar libres los números comprendidos entre 5000 y 6000, por si la zona anterior tiene que expandirse.

Más adelante, cuando hablemos de la programación modular, indicaré algunas otras sugerencias que pueden ayudar a aumentar la legibilidad de los programas, para hacerlos más fáciles de entender y modificar, a pesar de las pocas posibilidades que da el lenguaje BASIC en esta dirección.

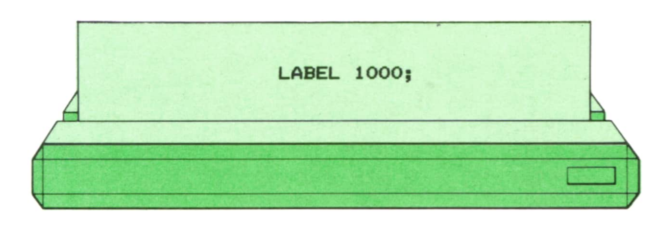


Etiquetas en PASCAL

Como se ha dicho a menudo, PASCAL es un lenguaje basado primordialmente en la programación estructurada, que mira con malos ojos la instrucción de transferencia y, consiguientemente, casi no tiene necesidad de etiquetas. Por esta razón, los programas PASCAL están normalmente desprovistos de ellas. Sin embargo, PASCAL es un lenguaje de programación completo y la instrucción de transferencia, aunque desaconsejada, existe. Por tanto, también ha de ser posible poner etiquetas a ciertas instruccio-

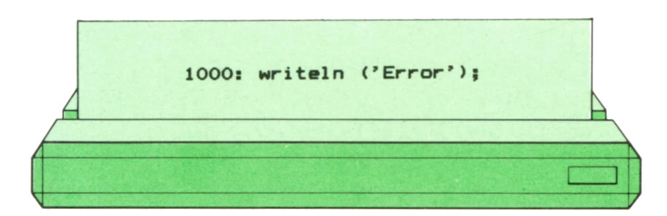
nes (muy pocas) que puedan ser alcanzadas directamente desde cualquier otro punto del programa. Esto suele aplicarse cuando en el transcurso de la ejecución del programa aparecen condiciones de error que hay que señalar y que deben dar lugar a una detención completa de su funcionamiento.

En principio, una etiqueta en PASCAL es, al igual que en BASIC, un número de instrucción (un número entero, positivo, de cuatro cifras como máximo). Al igual que los identificadores o variables, las etiquetas hay que definirlas, lo que se hace en la parte declarativa del programa con la instrucción siguiente:



donde la palabra reservada LABEL significa «etiqueta» en inglés.

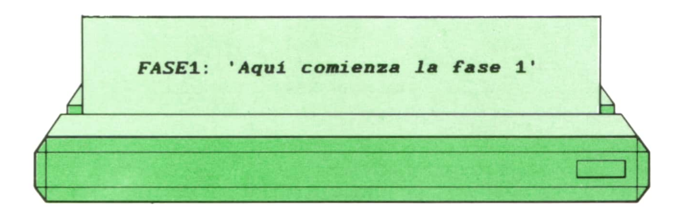
Además, la etiqueta se colocará a la izquierda de la instrucción correspondiente, separada de ella por dos puntos:



En algunos compiladores de PASCAL se permite utilizar nombres, además de números, como etiquetas de instrucciones.

 Etiquetas en APL

En el lenguaje APL las etiquetas son muy frecuentes, aunque no es obligatorio en modo alguno que cada instrucción tenga su etiqueta. Una etiqueta es un identificador, parecido en todo al nombre de una variable, y se coloca a la izquierda de la instrucción a la que da nombre, separada de ella por dos puntos:



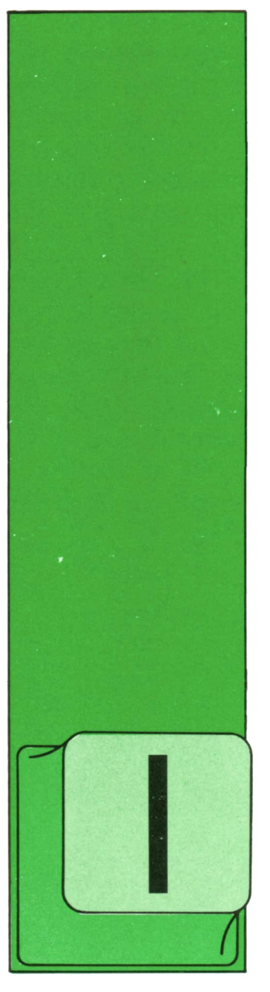
En el ejemplo anterior, cuando el programa alcance la instrucción de etiqueta FASE1 escribirá el mensaje: «Aquí comienza la fase 1».

La utilización de nombres tiene una gran ventaja sobre la de números de instrucción: que el nombre mismo de la etiqueta puede dar una idea de en qué parte del programa nos encontramos ayudando a hacerlo más legible.

APLICACIONES

HOJAS DE CALCULO LOTUS 1-2-3 (parte 2)

Más comandos de la hoja electrónica



IMPRIMIR se utiliza para la impresión de una hoja de trabajo. Permite establecer parámetros como: rango, márgenes, cabecera, pie de página, bordes, impresora a utilizar...

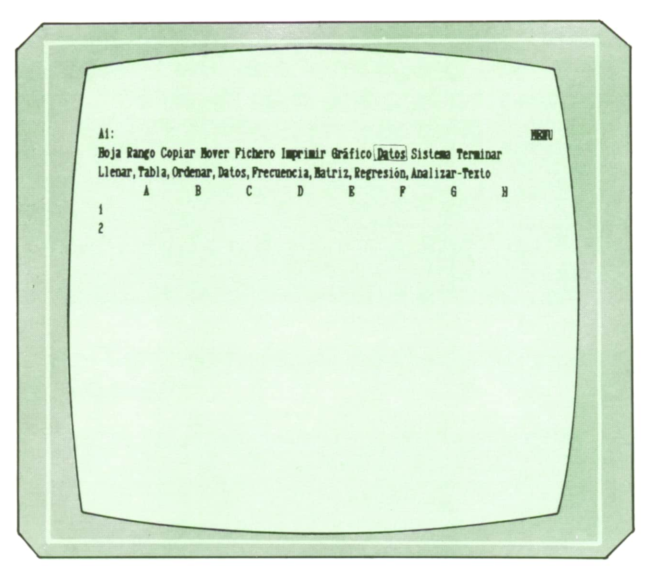
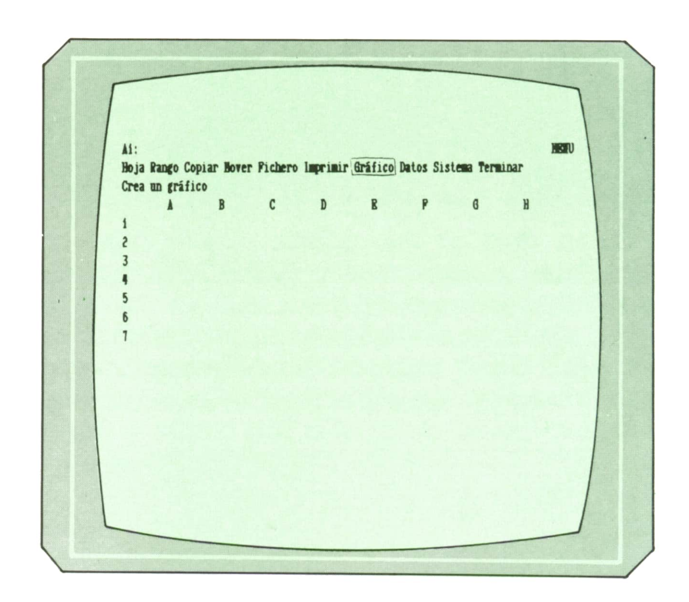
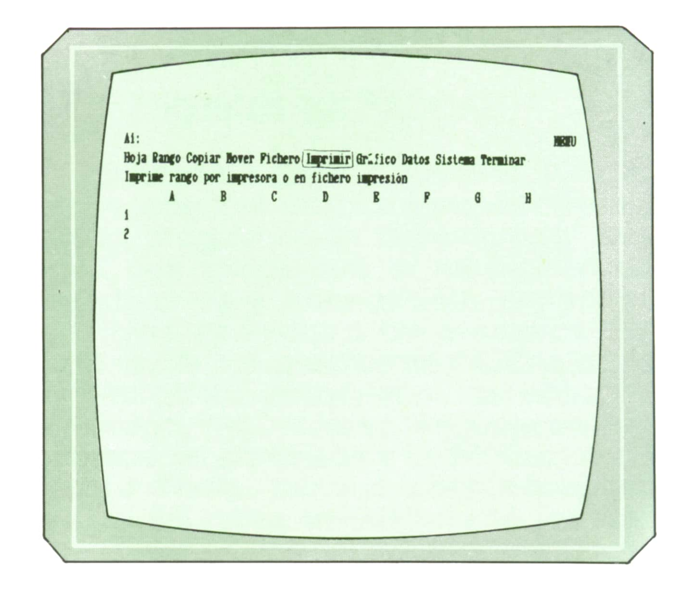


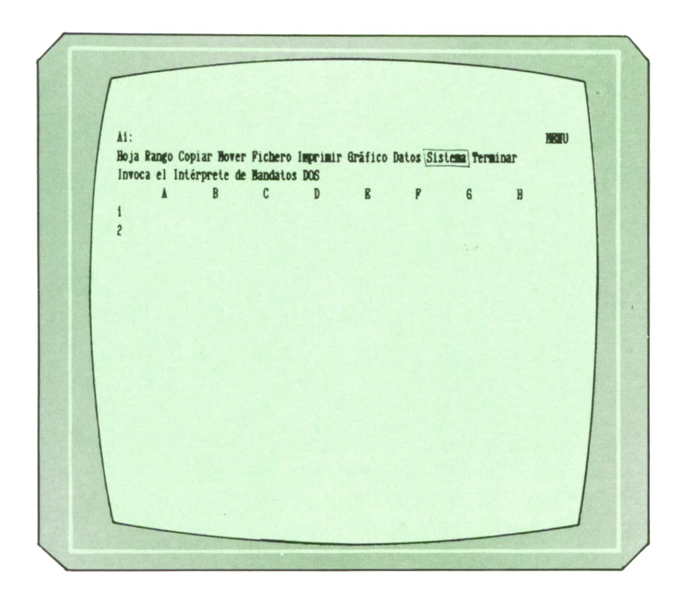
GRAFICO permite realizar un gráfico con un rango de valores de la hoja según cinco tipos diferentes: líneas, barras, sectores, barras apiladas, XY.



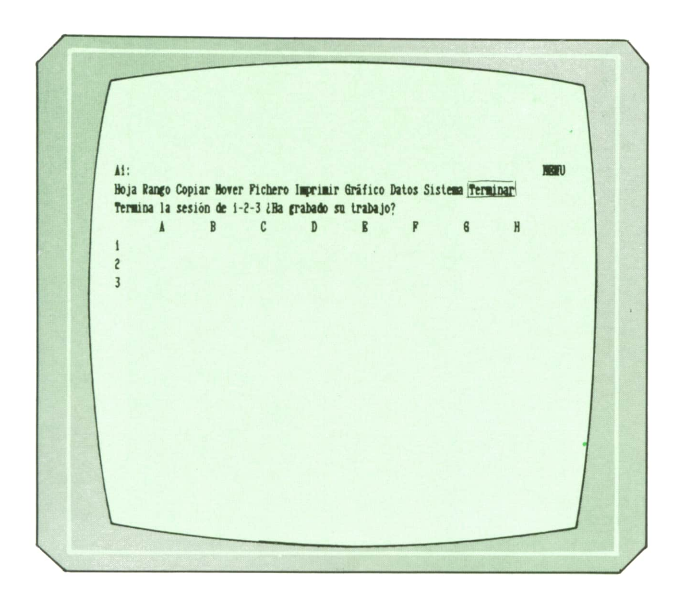
DATOS es el comando de base de datos.



SISTEMA permite salir temporalmente al DOS sin abandonar 1-2-3.



TERMINAR se utiliza para finalizar una sesión de trabajo. Recuerde que debe grabar su trabajo para no perderlo.



Las funciones

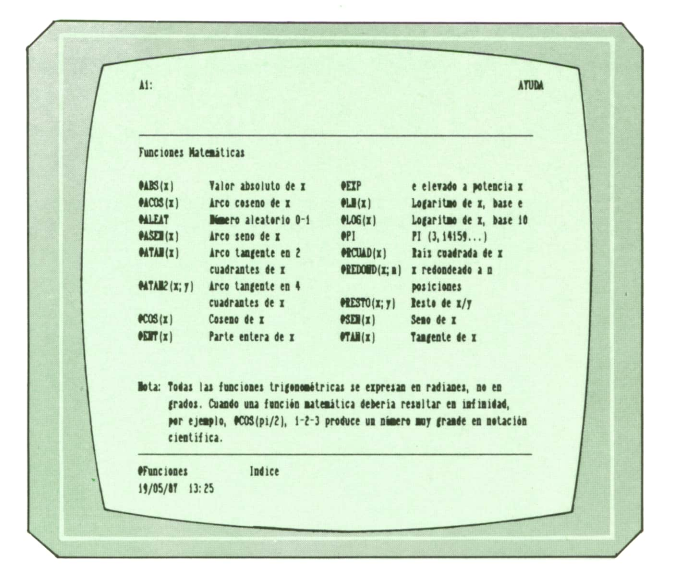
Queda por añadir algo sobre estas funciones que se encuentran en la gran mayoría de las hojas del mercado. El extendernos en el caso de las funciones del Lotus puede ayudar a conocer las del resto de las hojas de cálculo, ya que la mayor parte de ellas son estándar y las que aparecen de forma especial en Lotus pueden encontrarse en otro formato.

Una función es una fórmula que permite realizar cálculos de una forma fácil y rápida. En Lotus 1-2-3 se llaman funciones simplemente porque todas comienzan con dicho carácter. La mayoría de ellas constan de dos partes fundamentales:

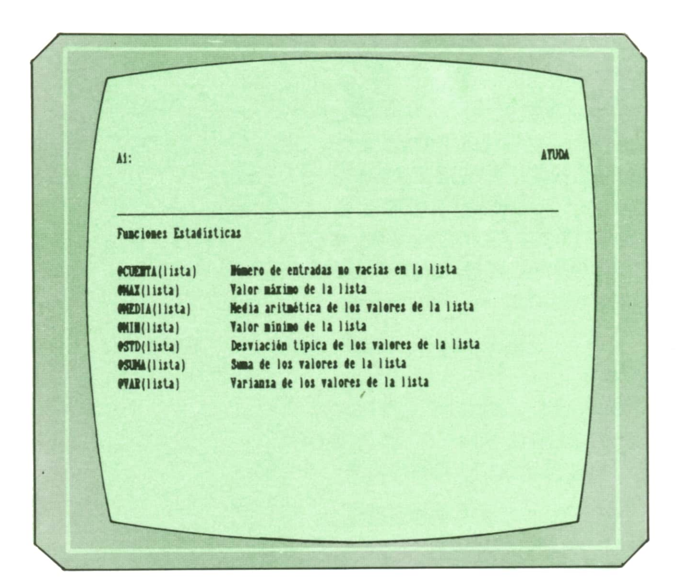
— Seguido del nombre de la función.
— El argumento o argumentos encerrados entre paréntesis y separados por puntos y comas.

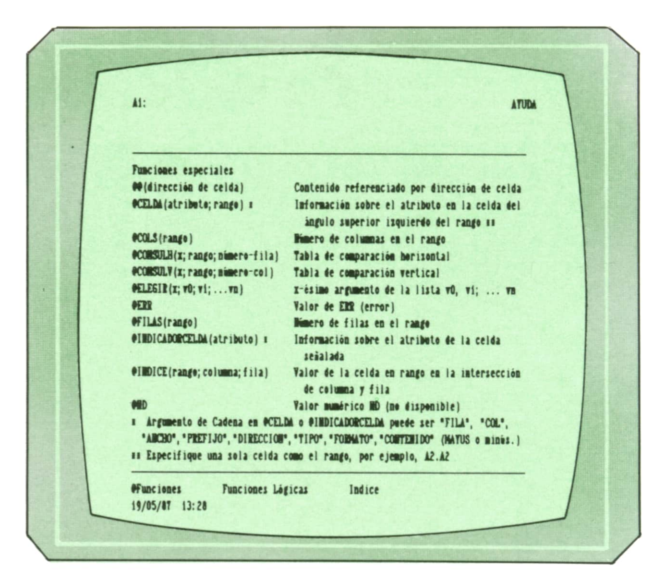
Se dividen en varios tipos:

— Funciones matemáticas: procesan valores numéricos y proporcionan resultados numéricos.



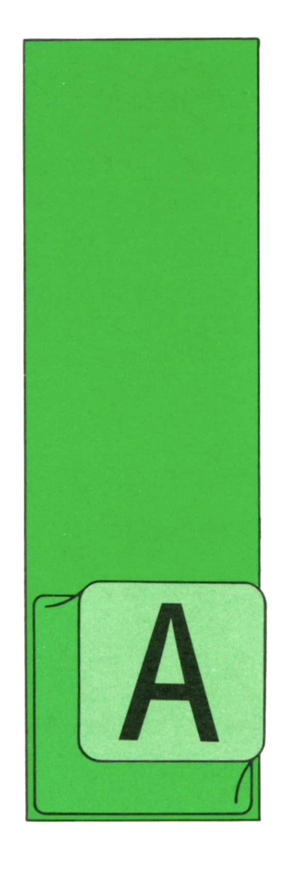
— Funciones financieras: calculan los resultados aplicables a préstamos, anualidades y amortizaciones.
— Funciones estadísticas: a partir de una lista de argumentos proporcionan un resultado de un parámetro estadístico.
— Funciones de fecha y hora.
— Funciones que generan números de serie.
— Funciones que aceptan números de serie.
— Funciones de cadenas: sus argumentos pueden ser numéricos o alfanuméricos.
— Funciones lógicas: son funciones numéricas cuyos resultados son 0 (FALSO) o 1 (VERDAD).
— Funciones especiales.





Con todas estas funciones se puede trabajar en la hoja y con algunas de ellas en la base de datos. Se utilizan dentro de las fórmulas, debe tenerse en cuenta que si el número de parámetros es erróneo o la sintaxis no es correcta Lotus se quejará con un ligero pitido indicándonos que debemos corregirlo.

PASCAL





Registros variantes

A veces, el aspecto de un registro depende del valor de uno de los campos. Por ejemplo, en las fichas de personal de una empresa podrían estar los datos del cónyuge sólo si el estado civil fuera casado. En PASCAL es posible definir fichas con campos fijos y con otros que pueden variar según el valor de alguno de aquéllos. Veamos un ejemplo:

```
type
 Fecha_t = record
             DiaMes : 1..31;
             Anyo : integer;
             Mes : (Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio,
                    Agosto, Septiembre,Octubre,Noviembre,Diciembre);
           end;

 Estado_t = (Soltero,Casado,Divorciado,Viudo);
 Nombre_t = array [1..20] of char;

 Ficha_t  = record
              Nombre: Nombre_t;
              Edad  : 16..70;
              case EstadoCivil: Estado_t of
                Soltero:    ();
                Casado :    (Conyuge   : Nombre_t;
                             FechaBoda: Fecha_t );
                Divorciado,
                Viudo:      (Final: Fecha_t)
            end;
```

Con esta estructura, todas las fichas de tipo Ficha-t tienen los campos Nombre, Edad y EstadoCivil. Además, y según el valor de este último campo, tienen los campos Cónyuges y FechaBoda si es casado y el campo Final cuando es viudo o divorciado. Nótese que algunos de los tipos utilizados han sido definidos sobre la marcha al definir el registro mientras que otros lo han sido con anterioridad.

Los campos variables siempre deben estar al final del registro y a continuación del campo que decide la estructura (EstadoCivil en este caso), utilizándose algo

muy similar a la instrucción CASE, sólo que poniendo a continuación de cada posible valor, y entre paréntesis, la lista de campos específicos que le corresponden (al final de cada variante es posible poner, a su vez, otra definición de campos, variables, etc.).

Un nombre de campo no puede aparecer repetido en varias listas; por otra parte, como ya se habrá imaginado el lector, el campo que decide debe ser de algún tipo escalar o subrango.

Realmente, no es obligatorio que exista un campo discriminante para tener definiciones alternativas de una ficha; un registro así se define exactamente igual que los del tipo que hemos visto pero omitiendo el discriminante, aunque, como se tiene que utilizar la «estructura» CASE, habrá que utilizar algún tipo escalar para señalar las diferentes variantes:

```
type
 Variantes_t = (Variante_A, Variante_B, Variante_C);

 Ejemplo_t = record
             CampoComun1: Tipo1;
             CampoComun2: Tipo2;

             case Variantes_t of
                Variante_A : (Campo_A1: Tipo_A1;

                              Campo_A2: Tipo_A2);

                Variante_B : (Campo_B1: Tipo_B1;
                              Campo_B2: Tipo_B2);

                Variante_C : ()   (* Ningún campo *)
             end;
```

Normalmente, los distintos campos de un registro ocupan posiciones consecutivas al ser almacenados en la memoria del ordenador, por lo que, si representamos con cajas las distintas porciones de ésta, una ficha del tipo Ejemplo-t podría ser:

Campo 1	Campo2	Campo.A1	Campo.A2

Sin embargo, nada nos impide que utilicemos la variante B, por lo que esa misma ficha también sería:

Campo 1	Campo 2	Campo.B1	Campo.B2

En otras palabras, estamos utilizando las mismas posiciones de memoria pero vistas con diferentes «ojos»; estas estructuras que permiten acceder de diferentes maneras a una misma zona de memoria son lo que se denomina «uniones libres» y, en principio, sólo resultan útiles para los programadores muy expertos. No obstante, vamos a ver algunas aplicaciones típicas:

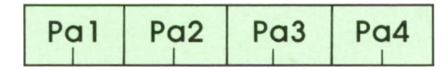
Como sabemos, todos los datos con los que tratan los ordenadores se guardan internamente utilizando una cierta cantidad de bytes o palabras de memoria; por ejemplo, un entero suele guardarse utilizando una palabra (dos bytes), y un ARRAY OF CHAR necesita tantos bytes como caracteres. A veces se necesita poder utilizar individualmente esas porciones de memoria, y ello es posible utilizando una unión libre:

```
record
   case boolean of
      true : ( Número: real);
      false: ( Pa1,
               Pa2;
               Pa3;
               Pa4 : integer)
end;
```

Suponiendo que un número real ocupase ocho bytes, una ficha así definida quedaría:

Número real (8 bytes)

Pero de esa misma ficha sería también:



De esta manera, podríamos saber cómo se guarda internamente un número real sin más que guardarlo en el campo Número de una ficha y leer los valores de los campos enteros. Como ejercicio, el lector podría escribir un programa e investigar cómo se guardan en memoria datos de tipo real, boolean, algún tipo escalar, etc.

Si el campo de un registro fuese una tabla de números y algún elemento en concreto de esa tabla se tuviera que utilizar mucho para hacer determinados cálculos, se podría ahorrar tiempo si nos evitásemos emplear un índice cada vez que se hiciera referencia a la variable:

```
record
 case boolean of
  true :
   (Tabla: array (1..10) of real);
  false:
   (T1,T2: real)
end;
```

De esta manera, T1 coincide con T(1) y T2 con T(2).

Normalmente, y por motivos derivados del hecho de ser un lenguaje compilado, es posible tener uniones libres también cuando existe el campo discriminante, con lo que, independientemente del valor de éste, se puede utilizar la variante que se desee; depende del programador el utilizarlo correctamente.

Vamos a preparar un procedimiento que presente el valor de los diferentes campos de las fichas del primer ejemplo:

```
procedure PonFecha (F: Fecha_t);
 (* Presenta la fecha con números: 23-11-86 etc. *)

 begin
  with F do
    write (DiaMes :2,'-',ord (Mes) + 1 :2,'-',Anyo mod 1900 :2)
 end;

(*-------------------------------------------------------------*)
procedure Presenta (Ficha: Ficha_t);
 (* Presenta las diferentes partes de una ficha *)

begin
with Ficha do
 begin
  writeln ('Nombre:        ',Nombre);
  writeln ('Edad:          ',Edad);
  write   ('Estado civil: ');

  case EstadoCivil of    (* esto es un CASE de verdad *)

    Soltero: writeln ('Soltero');

    Casado : begin
              writeln ('Casado');
              writeln ('Nombre del cónyuge: ',Conyuge);
              write   ('Fecha matrimonio  : ');
              PonFecha (FechaBoda);
              writeln
             end;

    Divorciado: begin
                 writeln ('Divorciado');
                 write   ('Desde: ');
                 PonFecha (Final);
```

```
                          writeln
                        end;

        Viudo : begin
                  writeln ('Viudo');
                  write   ('Desde: ');
                  PonFecha (Final);
                  writeln
                end
      end  (* fin de CASE *)

    end  (* fin de WITH *)
  end; (* fin del procedimiento *)
```

En general, y aunque se tengan diferentes campos según el valor del discriminante, la porción de memoria ocupada por una ficha cualquiera es siempre la misma e igual a la necesaria para el caso en que se precise más, que en el ejemplo es cuando la ficha tiene los campos Nombre, Edad, EstadoCivil, Cónyuge y FechaBoda.

NOTA:
Algunas versiones de PASCAL existentes para ordenadores domésticos carecen de la disponibilidad de crear registros variantes y, menos aún, de crear uniones libres.



Otro ejemplo de registros

(Antes de empezar, una advertencia: para entender este ejemplo hace falta tener una formación previa en números complejos.)

Una forma muy cómoda de manejar números complejos es utilizar registros con campos para las partes real e imaginaria. Como ejemplo de esto, vamos a escribir un procedimiento para, dados dos números complejos, obtener su producto y guardarlo en la variable correspondiente:

```
type
 Complejo_t = record Re,Im: real end;

procedure Producto (A,B: Complejo_t; var Resultado: Complejo_t);
  (* Obtiene el producto de A por B y lo guarda en Resultado *)
begin
 with Resultado do
   begin
    Re := A.Re * B.Re - A.Im * B.Im;
    Im := A.Re * B.Im + A.Im * B.Re
   end
end;
```

Como A y B y Resultado son del mismo tipo, sólo se puede utilizar un WITH (realmente, sería más eficiente usarlo con A o B, pues a sus campos se les hace referencia más veces que a los de Resultado).

Un método todavía más flexible sería utilizar variantes para poder emplear complejos en forma polar o binómica.

```
type
 Complejo_t = record
                case Forma: (Polar, Binomio) of
                 Polar: (Modulo, Argumento : real);
                 Binomio: (Re, Im: real)
          ıcto   end;

procedure Producto (A,B: Complejo_t; var Resultado: Complejo_t);

   (* Obtiene el producto de A por B y lo guarda en Resultado. *)
   (* Pasa todo a forma binómica antes de hacer los cálculos.  *)
   (* Resultado queda según el valor previo de su campo Forma. *)

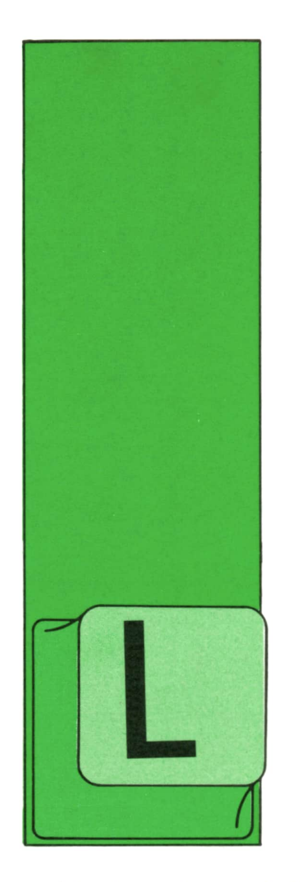
 var A1,B1,Res1: Complejo_t;
begin
 (*--------  Obtener copia binómica de A ---------*)
 with A do
   if Forma = Binomio then A1 := A
     else
      begin
       A1.Forma:= Binomio;
       A1.Re    := Modulo * cos (Argumento);
       A1.Im    := Modulo * sin (Argumento)
      end;
 (*--------  Obtener copia binómica de B ---------*)
 with B do
   if Forma = Binomio then B1 := B
     else
      begin
       B1.Forma:= Binomio;
       B1.Re    := Modulo * cos (Argumento);
       B1.Im    := Modulo * sin (Argumento)
      end;
 (*--------          Calcular          ---------*)
 with Res1 do
   begin
    Forma:= Binomio;
    Re   := A1.Re * B1.Re - A1.Im * B1.Im;
    Im   := A1.Re * B1.Im + A1.Im * B1.Re;

    (*--------   Guardar resultado      ---------*)
    if Resultado.Forma = Binomio then Resultado := Res1
      else
        begin
         Resultado.Modulo    := sqrt (sqr(Re) + sqr(Im));
         Resultado.Argumento:= arctan (Im/Re)
        end
   end (* with *)
end;
```

Este procedimiento quedaría más claro si se tuviesen procedimientos aparte para cambiar la forma de los números.

Pruebe el lector a hacer un programa «calculadora» para números complejos.

OTROS LENGUAJES





Lectura de ficheros

LA instrucción para leer un registro de un fichero secuencial es:

READ nombre-fichero
AT END sentencia

Al ejecutarse esta instrucción, se leerá un registro del archivo especificado en nombre-fichero. Cuando se hace un OPEN a un fichero, la cabeza de lectura/escritura se coloca en el primer registro del mismo. En un fichero secuencial, no se pueden releer registros. Es decir, si se ha leído el registro siete no se puede volver atrás y leer el seis. La lectura sólo es hacia delante.

Igualmente, si se ha leído el registro siete, para leer el número diez ha de pasarse por todos los anteriores: ocho y nueve. Así se irán leyendo uno a uno todos los registros. El final de un fichero se detecta automáticamente y en ese momento se ejecutan las sentencias que siguen al AT END, hasta encontrar el primer punto.

Es labor del programador codificar las instrucciones precisas para que, una vez finalizado un fichero, no se intenten leer más registros, evitando con ellas un error de ejecución.



Grabación en impresora

El lector ya conoce cómo se declaran ficheros asignados a impresora, que son listados en papel. Sólo pueden ser de escritura.

El formato de las instrucciones de impresión es diferente del de grabar en disco:

```
                                        { AFTER  }   { número }
WRITE nombre-registro FROM campo        {        }   {        }
                                        { BEFORE }   { PAGE   }
```

Nombre-registro es el definido en la FD. La opción FROM campo equivale a un MOVE (MOVE CAMPO TO nombre-registro) y un WRITE consecutivos (WRITE nombre-registro).

Se puede elegir entre dos partículas:

— AFTER: después.
— BEFORE: antes.

seguidas de un número entero o del literal PAGE.

La utilidad de estas sentencias se ve en los siguientes ejemplos:

WRITE LINEA AFTER 3

Imprimirá el contenido de LINEA después de dejar tres líneas en blanco.

WRITE LINEA BEFORE 3

Contrariamente a la anterior, escribirá primero y saltará después.

WRITE LINEA AFTER PAGE

PAGE indica salto de página. La impresora saltará hasta el comienzo de la siguiente página y luego escribirá.

Todas estas opciones dan una gran flexibilidad al programador a la hora de elaborar un informe.

Se presenta a continuación un programa que obtiene un listado por impresora del fichero de personal creado con anterioridad.

Este informe debe llevar una cabecera por cada página que se escriba, un contador de páginas y no se deben escribir más de 50 líneas en cada hoja.

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. EJ-FINAL.

ENVIRONMENT DIVISION.

 INPUT-OUTPUT SECTION.

   FILE-CONTROL.

        SELECT PERSONAL ASSIGN TO DISK
                        ORGANIZATION LINE SEQUENTIAL.

        SELECT SALIDA   ASSIGN TO PRINTER.


DATA DIVISION.

 FILE SECTION.

  FD PERSONAL   LABEL RECORD STANDARD
                VALUE OF FILE-ID 'PERSO.DAT'.

  01 REG-PERSONAL.
     05 NOM-APE                    PIC X(35).
     05 DIRECCION                  PIC X(25).
     05 TELEFONO                   PIC X(11).

  FD SALIDA     LABEL RECORD OMITTED.

  01 LINEA                         PIC X(80).


 WORKING-STORAGE SECTION.

  01 FIN                           PIC X            VALUE 'N'.

  01 CABECERA1.
     05 FILLER                     PIC X(30)        VALUE SPACES.
     05 FILLER                     PIC X(19)        VALUE
                 'LISTADO DE PERSONAL'.
     05 FILLER                     PIC X(21)        VALUE SPACES.
     05 FILLER                     PIC X(5)         VALUE 'PAG: '.
     05 PAGINA                     PIC 9(3)         VALUE ZERO.

  01 CABECERA2.
     05 FILLER                     PIC X(2)         VALUE SPACES.
     05 FILLER                     PIC X(18)        VALUE
                'NOMBRE Y APELLIDOS'.
     05 FILLER                     PIC X(20)        VALUE SPACES.
     05 FILLER                     PIC X(9)         VALUE
                'DIRECCION'.
     05 FILLER                     PIC X(18)        VALUE SPACES.
     05 FILLER                     PIC X(8)         VALUE
                'TELEFONO'.

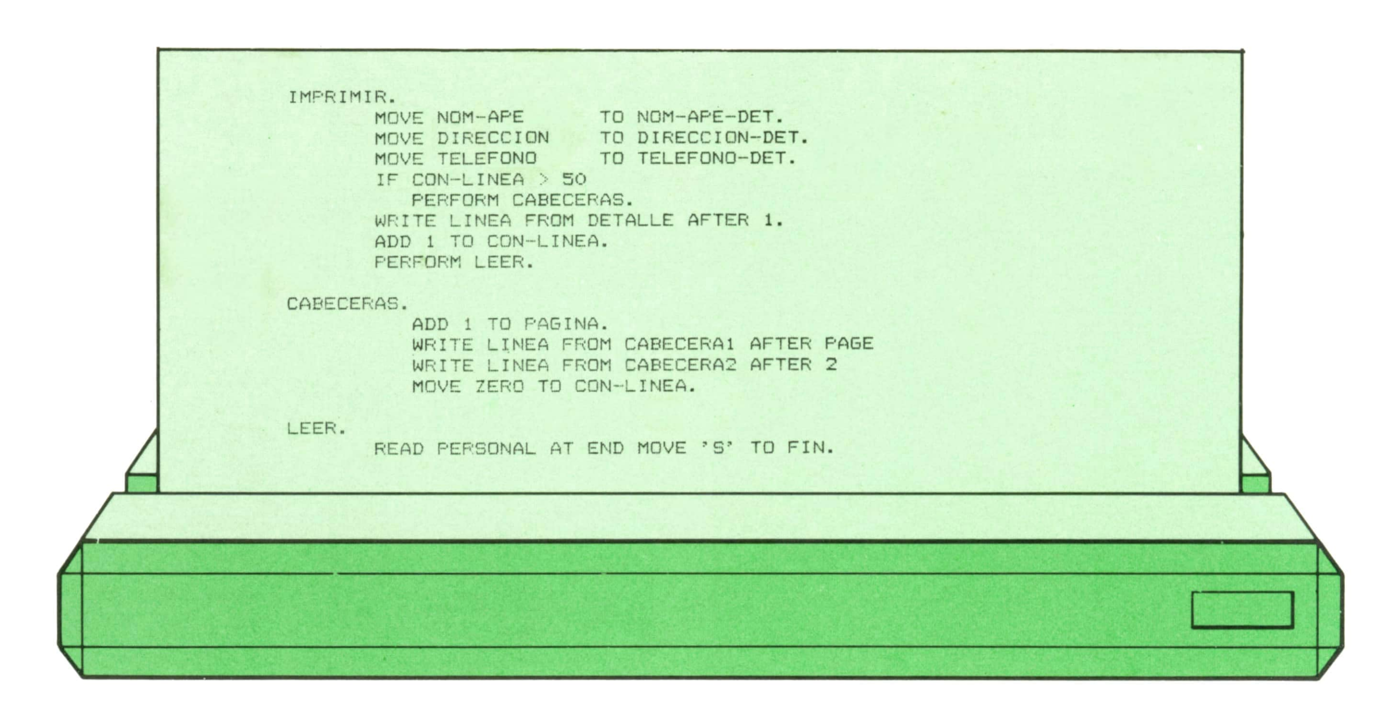
  01 DETALLE.
     05 FILLER                     PIC X(2)         VALUE SPACES.
     05 NOM-APE-DET                PIC X(35).
     05 FILLER                     PIC X(3)         VALUE SPACES.
     05 DIRECCION-DET              PIC X(25).
     05 FILLER                     PIC X(2)         VALUE SPACES.
     05 TELEFONO-DET               PIC X(11).

     01 CON-LINEA                      PIC 99           VALUE 51.


  PROCEDURE DIVISION.

  INICIO.
         OPEN INPUT  PERSONAL.
         OPEN OUTPUT SALIDA.
         PERFORM LEER.
         PERFORM IMPRIMIR UNTIL FIN = 'S'.
         CLOSE PERSONAL
               SALIDA.

  FIN-PROGRAMA.
         STOP RUN.
```



```
IMPRIMIR.
        MOVE NOM-APE        TO NOM-APE-DET.
        MOVE DIRECCION      TO DIRECCION-DET.
        MOVE TELEFONO       TO TELEFONO-DET.
        IF CON-LINEA > 50
           PERFORM CABECERAS.
        WRITE LINEA FROM DETALLE AFTER 1.
        ADD 1 TO CON-LINEA.
        PERFORM LEER.

CABECERAS.
           ADD 1 TO PAGINA.
           WRITE LINEA FROM CABECERA1 AFTER PAGE
           WRITE LINEA FROM CABECERA2 AFTER 2
           MOVE ZERO TO CON-LINEA.

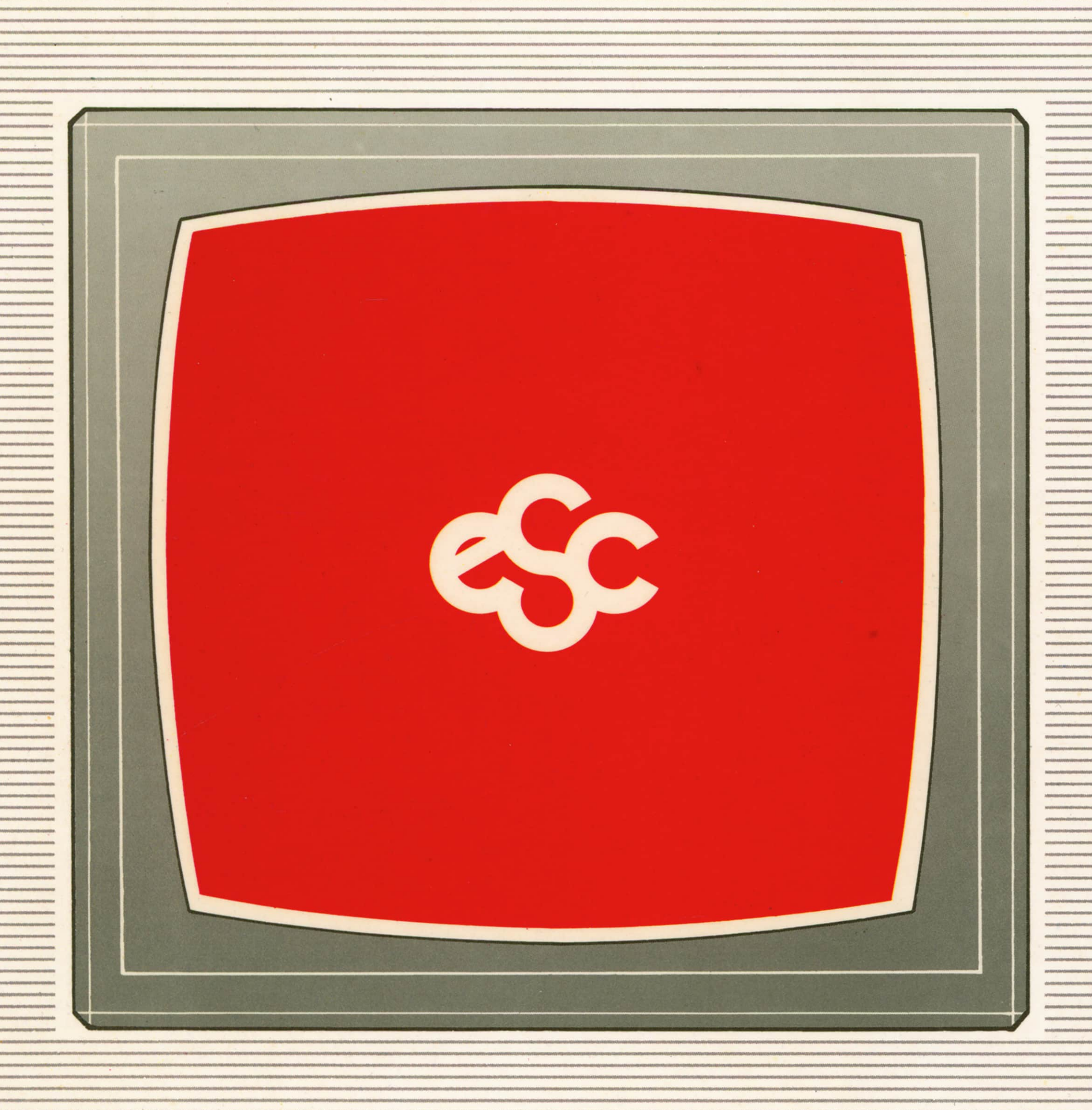
LEER.
        READ PERSONAL AT END MOVE 'S' TO FIN.
```

En la WORKING de este programa se describe FIN-FICHERO, que se utiliza para detectar el final del fichero y otro campo para contar el número de líneas escritas. El primero se inicializa con «N» y el segundo con 51.

A continuación se definen las cabeceras que van a aparecer en el listado y la línea de detalle.

Se abren los ficheros y se lee el primer registro de PERSONAL. Seguidamente se establece un bucle que se ejecutará hasta que se haya recorrido el fichero en su totalidad. Por cada registro leído se genera una línea en la impresora.

El salto de página se controla con la variable CONT-LINEA. Cuando ésta sea mayor de 50, se incrementa el contador de página, se inicializa el contador de líneas y se escriben las cabeceras.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼